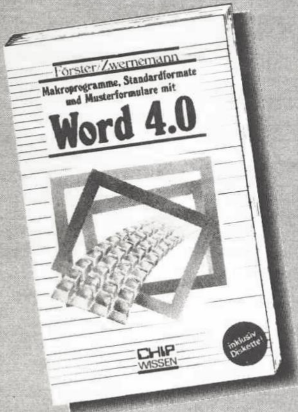
CHIP PLUS

EINE BEILAGE
IN CHIP NR. 4
APRIL 1990

PARALLEL RECHNER

CHIP WISSEN

Die kompetente Buchreihe rund um den PC



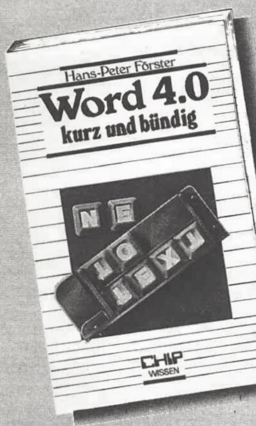
Hans-Peter Förster/Martin Zwernemann:

Makroprogramme, Standardformate und Musterformulare mit Word 4.0

144 Seiten, 42 Bilder, Hardcover
48,– DM (mit 5,25″-Diskette)
ISBN 3-8023-**0246**-X

Jeder der mit Word 4.0 Standardformulare ausfüllt, Serientexte schreibt und Etiketten bedruckt oder eigene Makroprogramme und Druckformatvorlagen erstellen möchte, braucht dieses Buch mit Diskette.

Auf der Diskette befinden sich nahezu alle Etikettenformate, Formate für Endlospapiere und Standardformulare.
Die Makroprogrammierung, das Arbeiten mit Druckformaten und Serientexten wird Schritt für Schritt anhand von Praxis-Beispielen erklärt.



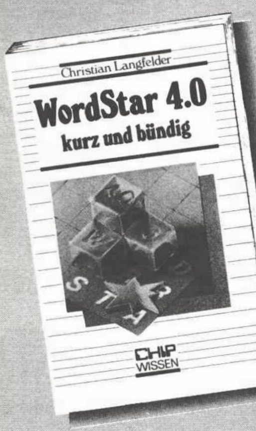
Hans-Peter Förster/Martin Zwernemann:

Word 4.0 kurz und bündig

256 Seiten, 67 Bilder, Hardcover
43,– DM/ISBN 3-8023-**0215**-X

Dieses Arbeitsbuch richtet sich an interessierte Einsteiger sowie an Word-4.0-Anwender, die schnell den Umgang mit dem komfortablen Textverarbeitungsprogramm erlernen wollen. Es ist so aufbereitet, daß es auch als Begleitmaterial für Word-4.0-Schulungen geeignet ist. Aber auch zum Selbststudium ist es in Ergänzung zum Word-4.0-Handbuch ein Hilfsmittel, das viele Tips und Tricks verrät.

* Wichtige Grundfunktionen
* Texte erstellen, korrigieren, speichern
* Texte laden, korrigieren, gestalten
* Texte direkt/indirekt formatieren
* Ausdruck
* Textbausteine
* Rechtschreibprogramm u.a.

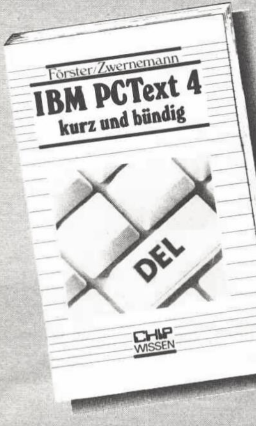


Christian Langfelder:

WordStar 4.0 kurz und bündig

184 Seiten, 46 Bilder, Hardcover
35,– DM/ISBN 3-8023-**0219**-2

Dieses Buch verfolgt einerseits das bewährte Konzept seines Vorgängers „WordStar kurz und bündig" und macht den Leser andererseits mit den neuen Eigenschaften und Möglichkeiten von WordStar 4.0 bzw. den Zusatzprogrammen von WordStar 4.0 Extra vertraut. Es ist somit für Einsteiger in der Textverarbeitung als auch für Umsteiger auf die neue Version bestens geeignet.



Hans-Peter Förster/Martin Zwernemann:

IBM PCText 4 kurz und bündig

200 Seiten, 47 Bilder, Hardcover
38,– DM/ISBN 3-8023-**0247**-8

Ziel dieses Buches ist es, dem Anwender des IBM Textverarbeitungsprogrammes PCText 4 einen Ratgeber zur Seite zu stellen, der alle Funktionen und Befehle knapp aber präzise vorstellt.

Immer wieder wird der Leser beim Durcharbeiten des Buches zur direkten Eingabe der notwendigen Tasten aufgefordert. Die Knappheit des Textes und die Fülle der Informationen machen das Buch zu einem wichtigen Nachschlagewerk bei der täglichen Arbeit und damit zu einer praktischen Ergänzung zum IBM-Handbuch.

Haben Sie schon den neuen „CHIP-Katalog"?
Bestellen Sie gleich!

VOGEL Buchverlag Würzburg
Postfach 6740 · 8700 Würzburg 1

Parallel ist nicht gleich parallel

Ulrich Parthier

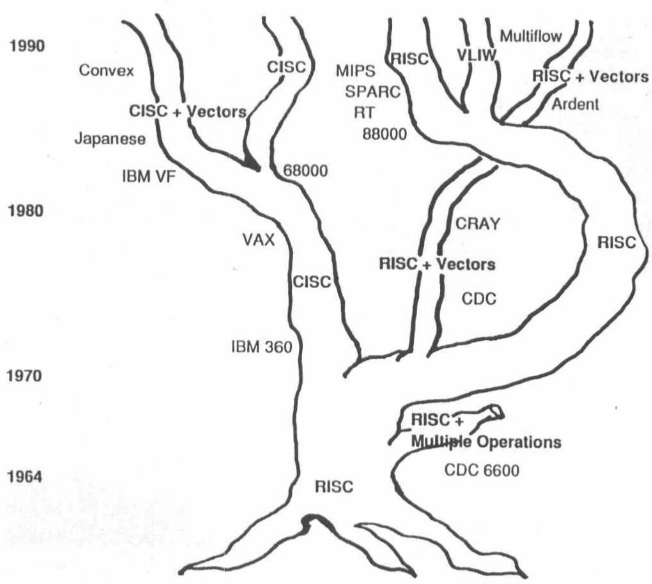
In den vergangenen Monaten ist die DV-Welt von Ankündigungen im Bereich der Parallelrechner geradezu überschwemmt worden. Systemerweiterungen „alter" Parallelrechner, Rechner mit neuen Konzepten und Weiterentwicklungen aus dem universitären Bereich machen dem Anwender die Entscheidung schwer.

Der verstärkte Einzug der Datenverarbeitung in fast allen Bereichen unseres Lebens in den letzten zwanzig Jahren hat zu immer mehr Daten geführt, die – bevor sie gespeichert werden (ebenfalls ein großes Problem) –, zuerst einmal verarbeitet werden müssen. Die immer weiter voranschreitende Miniaturisierung der Speicherchips bei ihrer gleichzeitig wachsenden Leistungsfähigkeit hat uns bei der Bewältigung unserer Probleme sicherlich geholfen. Dennoch, ohne eine Weiterentwicklung der klassischen Von-Neumann-Architektur wäre der heutige Leistungsstand der Super- und Mini-Supercomputer nicht denkbar.

Diese Entwicklung und den aktuellen Stand der Technik versuchen wir, im folgenden aufzuzeigen. Natürlich können wir nicht über alle Strömungen und technischen Details berichten.

Interessant wird die Frage sein, wie die Akzeptanz des Marktes ist. Denn eines erscheint den Experten trotz steigenden Leistungsbedarfs klar: Für alle Rechner ist kein Platz.

Die Anwendungsgebiete, in denen die Parallelrechner Einzug gehalten haben, haben sich weiter ausgedehnt. Waren es früher vor allem technisch-wissenschaftliche Applikationen (etwa Simulation, Bildverarbeitung, Strömungsmechanik, Meteorologie, Luft- und Raumfahrt, Kfz-Entwicklung), so finden sich jetzt auch immer mehr kommerzielle Anwendungsgebiete, zum Beispiel bei der Online-Transaktionsverarbeitung (OLTP), die im Finanzsektor (Börse, Lotterie, Banken, Versicherungen) um sich greift. Das A und O des Erfolges wird aber weiterhin nicht in genialen Konzepten und ihrer adäquaten Umsetzung liegen, sondern in der Verfügbarkeit und der leichten Portierbarkeit der Software. Und das Parallelrechner auch nicht gleich Parallelrechner ist, zeigt unser Heft auf den folgenden Seiten. ■



Die Parallelrechner-Entwicklung

Abbildung: GEI Rechnersysteme

Entwicklung der Rechnerarchitekturen

Silvia Parthier



Die Autorin Silvia Parthier studierte an der Universität Regensburg Volkswirtschaftslehre. Seit 1985 arbeitet sie als freie DV-Journalistin, Autorin und DV-Beraterin in Eichenau bei München.

Das Thema Parallelrechner hat sich in der letzten Zeit zu einer Art Modethema in der EDV-Branche entwickelt. Es vergeht kaum ein Monat, in dem nicht ein weiterer Rechner mit neuem Konzept oder innovativen Elementen auf dem Markt erscheint.

Noch vor vier Jahrzehnten war die Rechnerwelt einfach und überschaubar, waren die Rechnerarchitekturen simpel. In der Folgezeit ging der Trend zu immer komplexeren und unüberschaubareren Architekturen und Befehlssätzen. Das Zeitalter der CISC-(Complex Instruction Set Computer)-Prozessoren begann.

Ergibt CISC einen Sinn?

Mit Computerprogrammen verhält es sich ähnlich wie etwa mit einem Haus, das aus kleinen Elementen, etwa Steinen, und größeren Einheiten, zum Beispiel Räumen, besteht. Operationen als kleinste Einheit haben als übergeordnete Strukturen Schleifen und Unterprogramme. Die Ausführung der Operationen obliegt der Hardware. Da Operationen nur relativ kleine Arbeitsschritte, beispielsweise „Lade x", „Speichere y" oder „Hole z in den Arbeitsspeicher", ausführen, benötigt man für komplexe Aufgabenstellungen sehr viel Zeit.

Die auf die ersten CISC-Rechner folgende Rechnergeneration verstand bereits komplexere Befehle. Ein Beispiel dafür wäre „Kopiere eine Zeichenkette von einem Speicherbereich in einen anderen". Die Hardware wurde also um Teile erweitert, die solche komplexeren Befehle in einem einzigen Programmbefehl ausführen konnten.

Diese Makroinstruktionen arbeitet der Rechner nach einer eingebauten Arbeitsanweisung – der Firmware – ab, die eine Reihe von Elementaroperationen auslöst.

Die Blütezeit der CISC-Rechner dauerte mehr als zwanzig Jahre. An der amerikanischen Renommieruniversität in Stanford schließlich fand man heraus, daß komplexe Befehle eigentlich genau das sind, was der Rechner nicht braucht, denn die meisten der Instruktionen wurden nur ganz selten genutzt. Die Ära der RISC-(Reduced Instruction Set Computer-)Geräte brach an.

Die Supercomputer

Im technisch-wissenschaftlichen Bereich spielten sich etwa zur gleichen Zeit ähnliche Vorgänge ab. Nur wurden die Systeme hier nicht um komplexere Instruktionssätze, sondern um ganz spezielle komplexe Instruktionen erweitert. Diese umfaßten die Arbeit mit Zahlenreihen, den Vektoren. Sie spielen in naturwissenschaftlichen Anwendungen eine wichtige Rolle. So entstanden Vektoreinheiten-Hardwarezusätze, die auf die schnelle Abarbeitung solcher Operationen spezialisiert sind. Der Nachteil bestand in der Unflexibilität der Vektoreinheiten.

In zunehmendem Maße ging man auch in der technisch-wissenschaftlichen Welt dazu über, dem Programmierer beziehungsweise dem Compiler die Elementarfunktionen der Hardware offenzulegen, damit die jeweils gewünschte Kombination selbst gewählt werden konnte. Vertreter dieser Kategorie innerhalb der RISC-Welt finden sich in Rechnern der Klassen LIW (Long Instruction Word) und VLIW (Very Long Instruction Word).

Mit der Rückbesinnung auf das RISC-Prinzip im technisch-wissenschaftlichen Bereich war allerdings mit der Parallelverarbeitung ein weiterer wichtiger Faktor zu berücksichtigen. Hohe Rechenleistung – das hatte man schnell bei allen Arten der bisher genannten Rechnerarchitekturen erkannt – lassen sich durch parallele Abarbeitung von Befehlen erzielen. In der Praxis gibt es jedoch unterschiedliche Grade der Parallelisierung. Das Problem bei der Parallelisierung liegt im wesentlichen in der Synchronisation der Arbeit, die bei komplexen und schwierigen Programmen um so komplizierter wird.

Das in der Welt der kommerziellen Anwendungen häufig anzutreffende Multiprocessing und das in Technik und Wissenschaft übliche Parallelprocessing wird zunehmend durch eine dritte Art der Parallelisierung abgelöst, die „feinkörnige Parallelisierung". Hier gab es von Control Data mit der CDC 6600 in den sechziger Jahren bereits Ansätze, die sich jedoch nicht durchsetzen konnten.

Die feinkörnige Parallelisierung

Mit der „Wiedererfindung" des RISC-Prinzips hat die eben erwähnte dritte Art der Parallelisierung – die man als „feinkörnige Parallelisierung" bezeichnet – neue Impulse erhalten. Ähnlich wie bei den RISC-Prozessoren war dies der Compilertechnik zu verdanken. Der Erfolg waren die LIW- und VLIW-Rechner. Die Grundidee besteht darin, die schwierige Aufgabe der Arbeitsteilung auf verschiedene Prozessoren durch das Übersetzungsprogramm, den Compiler, vorwegnehmen zu lassen. Dieser erzeugt ei-

nen Arbeitsplan mit genauen Anweisungen, welcher Prozessor welchen Befehl beim Programmlauf bearbeiten soll. Der Vorteil: Auch bei komplexen Synchronisationsvorgängen sind hohe Verarbeitungsgeschwindigkeiten möglich. Ein weiterer Vorteil besteht darin, daß bei traditionellen Vektorrechnern meist noch die Algorithmen auf die Rechner abgestimmt werden müssen. Diese Arbeit entfällt bei der neuen Rechnergeneration. Diese Rechner machen heute den Supercomputern mit Vektoreinheiten das Leben schwer. FPS und Multiflow sind Vorreiter dieser Technologie gewesen, und auch bei Unternehmen wie Intel, Mips, Sun oder Motorola haben diese Konzepte mittlerweile an Einfluß gewonnen.

Das VLIW-Konzept

Ein Beispiel für einen Rechner, der auf dem VLIW-Konzept basiert, ist der Supercomputer Trace von GEI Rechnersysteme in Aachen. Hinter VLIW verbirgt sich ein besonders langes Befehlswort. Weil mehrere Einzelkommandos in ein Befehlswort passen, wird für den Rechengang entsprechend weniger Zeit benötigt. VLIW ist somit eine besondere Form der Parallelverarbeitung, bei der die Aufgabe zur simultanen Verarbeitung in Einzelteile zerlegt und auf mehrere Prozessoren oder Rechenwerke eines Rechners verteilt wird.

In der Forschung ist schon mit Hunderten und Tausenden von Prozessoren zur Steigerung der Leistungsfähigkeit experimentiert worden. Abgesehen davon, daß es für dieses Leistungsspektrum nur wenig Anwendungsgebiete gibt, liegt der Nachteil in der Software. Sie ist meist nur für Einzelprozessoren geschrieben – entsprechend kompliziert und langwierig ist ihre Portierung.

Technologietransfer

Bei VLIW hingegen können solche Programm-Modifikationen mit spezieller Software automatisch durchgeführt werden. VLIW wurde in den USA an der Yale-Universität von Joseph Fisher entwickelt. Dessen Firma Multiflow, die die Trace-Systeme herstellt, ist ein gelungenes Beispiel für den Technologietransfer von der Universität in die Wirtschaft.

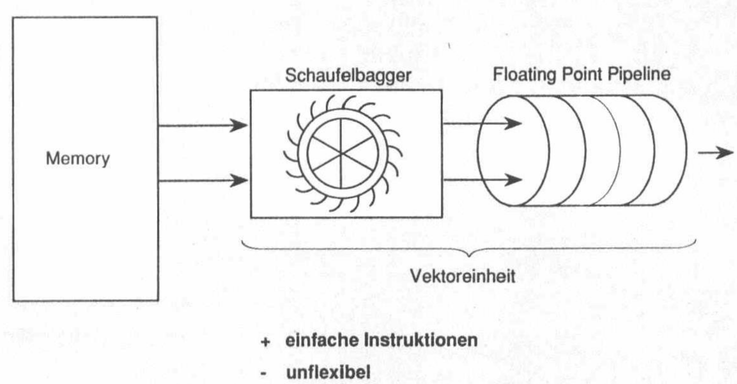
Während andere Computerhersteller ihren Erfolg in der Kopplung mehrerer CPUs suchen, arbeitet die Multiflow-Lösung mit einer Kombination preisgünstiger Hardware und einer innovativen Softwarelösung. Ein Compiler erlaubt es, bis zu 28 Befehle zusammenzufassen und gleichzeitig zu bearbeiten. Hinter dieser Methode verbirgt sich also ein ganz anderer Denkansatz als bei anderen Konzepten, wo etwa 64 000 Prozessoren einen Befehl gleichzeitig bearbeiten. Der Trace-Compiler faßt unterschiedliche Befehle so zusammen, daß möglichst viele gleichzeitig verarbeitet werden können.

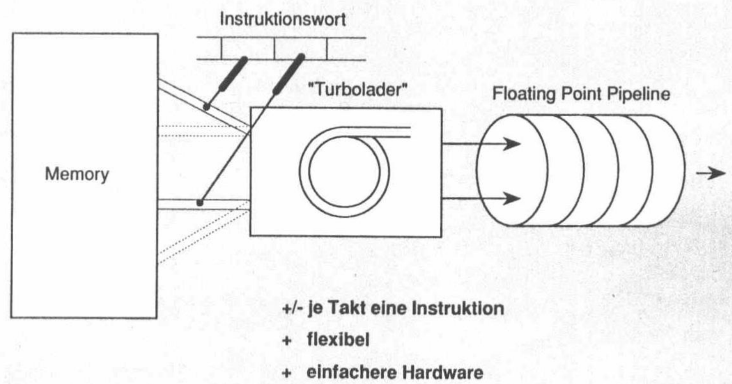
Der bedingte Sprung

Die wichtigste Neuerung im Trace-Compiler ist die Lösung des sogenannten bedingten Sprungs. Solche Sprünge sind Verzweigungen im Programm, die den Ablauf des Rechengangs aufgrund einer Entscheidung bestimmen. Zum Beispiel kann ein Programm den Rechner anweisen, einen bestimmten Programmpfad nur dann einzuschlagen, wenn bestimmte Bedingungen erfüllt sind, sonst wird ein anderer Pfad verfolgt. Weil ein „bedingter Sprung“ erst nach Vorliegen der Entscheidungskriterien bearbeitet werden kann, ist eine Parallelisierung eigentlich nicht möglich. Mit Hilfe neuartiger Algorithmen kann der Trace-Compiler nun die Sprungentscheidung vor Ablauf des Programms selbst ermitteln. Er kann daher den Pfad (daher der Name: Pfad, engl. trace) durch das Programm vorausbestimmen. Nun ist der Weg frei für die optimale Nutzung des Compilers. Er kann alle 28 Rechenwerke ohne Unterbrechung beschäftigen.

Auch die Carnegie-Mellon-Universität in den USA nutzt die Vorteile des VLIW-Konzepts. Von den Systemen dieser Kate-

Arbeitsprinzip eines Vektorrechners





Arbeitsprinzip eines VLIW-Rechners

Abbildungen: GEI Rechnersysteme

gorie erscheint die Multiflow-Entwicklung als die ausgereifteste. An der genannten Universität entstand auch Mach, eine vom US-amerikanischen Department of Defense (DoD) finanzierte Unix-Weiterentwicklung, die mittlerweile in den Next-Rechnern vom Apple-Mitbegründer Steve Jobs implementiert ist, und derzeit entwickelt dort ein Team unter der verantwortlichen Leitung von H. T. Kung einen VLIW-Rechner, der den Codenamen Warp trägt und vom General-Electric-Konzern demnächst vermarktet werden soll.

Die nächste Generation

Gemeinsam mit Intel arbeitet die Gruppe iWarp ebenfalls an einem VLIW-Rechner der nächsten Generation. Intel hat mit i860 ja einen Chip mit ganz schlichten VLIW-Eigenschaften herausgebracht, der unter bestimmten Bedingungen zwei Operationen zur gleichen Zeit ausführen kann. Ähnliche Eigenschaften werden von Motorola 68040 erwartet, dessen Einführung sich jedoch erheblich verzögert hat und der nun im Januar dieses Jahres offiziell vorgestellt worden ist. Auch die nächste Generation der Mips-RISC-Rechner soll über VLIW-Eigenschaften verfügen.

Zwar ist es jetzt noch zu früh zu sagen, welches Parallelrechnerkonzept sich durchsetzen wird – das wird letztendlich auch von den Applikationen abhängig sein. VLIW scheint für die Zukunft jedoch nicht die schlechtesten Karten zu haben. Zu dem ausgereiften Konzept kommt noch ein günstiges Preis-/Leistungsverhältnis hinzu.

Alternativen im Markt

Unter der Vielzahl der Parallelrechnerkonzepte können in dieser CHIP PLUS-Ausgabe natürlich nicht alle Platz finden. Einen Teil der Parallelrechner-Szene haben wir jedoch bereits in früheren CHIP PLUS-Ausgaben (3/88 und 8/88) vorgestellt. Dort wurde über den Stand des Suprenum-Rechners ebenso berichtet wie über die Konzepte der Cray- und Alliant-Systeme. Dem Transputer und seinen Implementierungen wie etwa der Supercluster-Serie wurde ein ganzes Heft gewidmet. Dem immer wichtiger werdenden Thema „Neuronale Netze" wird außerdem in Kürze eine ganze CHIP-PLUS-Ausgabe gewidmet sein. ■

SCO Unix Multiprocessor Extension

Silvia Parthier

Bei Unix handelt es sich traditionell um ein Multitasking- und Multiuser-Betriebssystem, das dazu dient, die Verarbeitungsleistung eines Computers mit einem einzelnen Prozessor an mehreren Arbeitsplätzen verfügbar zu machen.

Im Laufe der letzten Jahre sind jedoch immer mehr Computer auf dem Markt erschienen, bei denen zur Leistungssteigerung mehrere Prozessoren parallel arbeiten. Konsequenterweise wurden auch Multiprozessor-Versionen von Unix entwickelt, die die höhere Leistung von Multiprozessor-Rechnern ausnutzen können. Das Ziel der Anstrengungen war, den Durchsatz eines einzelnen Unix-Computers zu erhöhen, indem eine Vielzahl von Prozessoren unter der Kontrolle eines übergeordneten Betriebssystems zusammenarbeiten. Ein solcher Rechner verfügt – wenn die Unix-Implementierung entsprechend erfolgt – über eine skalierbare Leistung, das bedeutet, der Systemdurchsatz läßt sich durch einfaches Hinzufügen von Prozessoren erhöhen.

Das Betriebssystem Unix muß wesentlich modifiziert werden, um diesen Anforderungen gerecht zu werden. Diese Änderungen öffnen indes der Inkompatibilität zu bestehenden Standards Tür und Tor. Aus diesem Grund hat Santa Cruz Operation eine Multiprozessor-Erweiterung für SCO Unix auf den Markt gebracht, mit der die Vorteile eines Multiprozessor-Rechners bei voller Kompatibilität zum Standard nutzbar sind.

Bei der SCO Unix Multiprocessor Extension (SCO MPX) handelt es sich um ein Softwarepaket, das für zahlreiche Multiprozessor-Rechner wie beispielsweise Compaq Systempro, Zenith Z1000, Corollary SMP/386 und SMP/486 verfügbar ist.

Diese Systeme können unter einer normalen Version von SCO Unix System V/386 als Singleprozessor-Rechner betrieben werden. Soll die Systemleistung erhöht werden, fügt der Anwender weitere Prozessoren und für jeden Prozessor jeweils eine SCO Unix Multiprocessing Extension hinzu. Hierdurch entsteht eine Workstation, ein Server oder ein Mehrplatzrechner. Die Ausbaumöglichkeiten sind nahezu unbeschränkt.

Beim Hinzufügen einer Extension wird der Unix-Kernel jedesmal um ein Codestück erweitert, wodurch der jeweilige Prozessor vom Betriebssystem in die Verarbeitung integriert wird. Jeder Prozessor führt sowohl System- als auch Anwenderprogramme so aus, als stünde ihm das gesamte Rechnersystem zur Verfügung. Das bedeutet, auf allen Prozessoren laufen SCO Unix als Betriebssystem, User-Tasks und Einheitentreiber; alle Prozessoren haben Zugriff auf den Hauptspeicher. Wichtig dabei ist, daß sämtliche Prozesse gleichzeitig stattfinden können.

Diese Methode stellt somit eine Lösung für das altbekannte Problem dar, daß die CPU eines Rechnerns zum „Flaschenhals" für den Systemdurchsatz wird. Die Steuerung des Computers wird nun von einer symmetrischen Multiprozessor-Version des Unix-Kernel übernommen.

Jeder Prozessor greift dabei auf die standardisierten unveränderten Algorithmen im Unix-Kernel zu. Kernel- und System-Tasks werden vom nächsterreichbaren freien Prozessor bearbeitet, wobei das Multiprozessor-Scheduling über eine einfache Erweiterung vorgenommen wird. Die Verarbeitungslast wird bei diese Methode gleichmäßig auf die vorhandenen CPUs verteilt. Hierdurch ergibt sich ein optimaler Durchsatz für das Gesamtsystem.

Die Autorin Silvia Parthier
studierte an der Universität Regensburg Volkswirtschaftslehre. Seit 1985 arbeitet sie als freie DV-Journalistin, Autorin und DV-Beraterin in Eichenau bei München.

Das Innenleben des Compaq Systempro

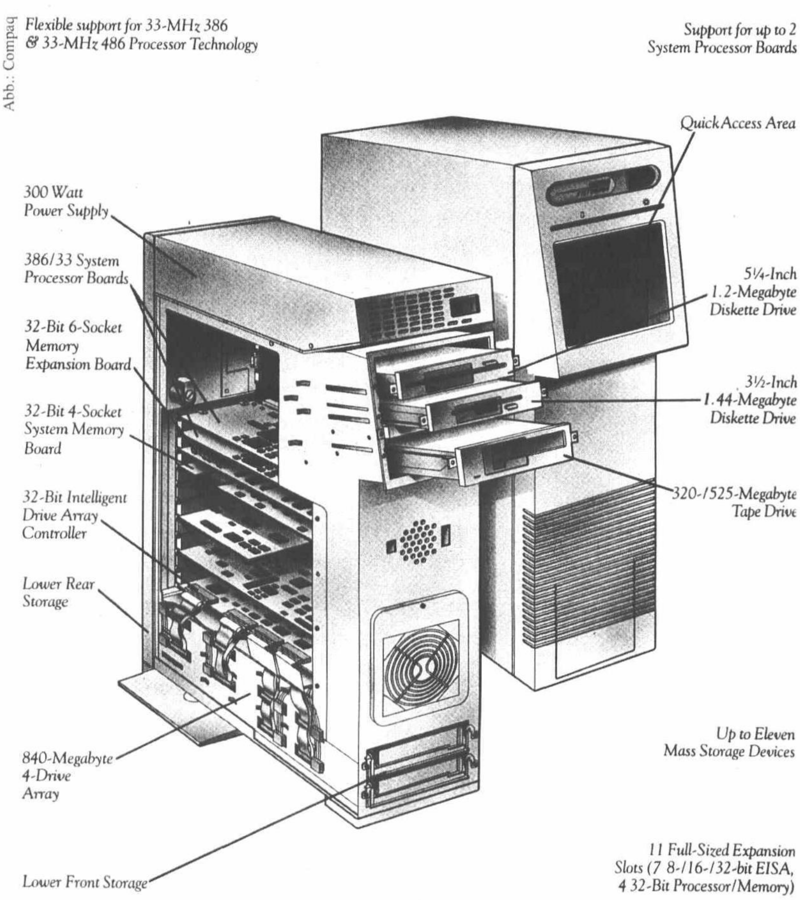


Abb.: Compaq

Gemeinsame Entwicklung

Santa Cruz Operation hat die Entwicklung der SCO Unix Multiprocessor Extension in enger Zusammenarbeit mit dem Hardware-Unternehmen Corollary vorgenommen.

Corollary hat einen Hochgeschwindigkeitsbus und ein spezielles Multiprozessor-Kartenset entwickelt und somit die Voraussetzungen für die Softwareerweiterungen geschaffen. Die Hardwarearchitektur von Corollary trägt die Bezeichnung ATtack. Sie verbindet mehrere eng gekoppelte Prozessoren mit gemeinsam genutztem Speicher über ein Bussystem mit hoher Übertragungsgeschwindigkeit.

Die Binärkompatibilität zu bestehenden Standards wird durch ein duales Bussystem erreicht. Während die Prozessoren untereinander und mit dem Speicher über einen 32-bit-Bus Daten austauschen, erfolgt die Kommunikation mit Peripheriegeräten über einen herkömmlichen AT-Bus. Dadurch können auch die Einheitentreiber von SCO für Peripheriegeräte ohne Änderungen eingesetzt werden. Hingegen erfolgt der Speicherzugriff innerhalb der ATtack-Architektur über den 32-bit-Bus und einem Write-back-Cache-Speicher mit 64 KByte im Direct-Mapped-Verfahren. Diese Architektur bildet die Hardwaregrundlage für die SCO Unix Multiprocessor Extension.

In der von Corollary vorgestellten Hardwarearchitektur und der von SCO entwickelten Multiprocessing-Erweiterung werden alle Prozessoren gleich behandelt. Das bedeutet, daß die Verarbeitungsleistung eines Rechners durch einfaches Hinzufügen einer CPU und einer SCO Unix Multiprocessor Extension erhöht werden kann.

Die SCO/Corollary-Lösung sieht vor, daß alle Prozessoren Zugriff auf den gesamten verfügbaren Speicher haben. Die CPUs können, wenn nötig, jederzeit direkt miteinander kommunizieren. Es besteht keine Notwendigkeit, bestimmte Programme bestimmten Prozessoren zuzuweisen. Vielmehr erhält jeder Prozessor vollständige Informationen über die Position der einzelnen Tasks im Speicher und kann auf diese direkt zugreifen. Gemeinsamer Speicher, direkter Speicherzugriff (DMA), Messages und Semaphoren sind durchweg auf eine hohe Verarbeitungsgeschwindigkeit ausgelegt.

Einheitentreiber benötigen in der Regel Zugriff auf den AT-Bus. Aus diesem Grund sorgt der Extension-Kernel dafür, daß Einheitentreiber durchweg auf dem Prozessor zum Laufen kommen, der den AT-Bus bedient.

Der Hintergrund: Ein Kernel-Modul, das nicht speziell für den Multithread-Betrieb vorgesehen ist, wird automatisch durch den Basisprozessor ausgeführt. Der Vorteil ist darin zu sehen, daß jedes Kernel-Modus-Programm ohne Änderungen auch in der Multiprocessing-Umgebung läuft. Dies betrifft neben Einheitentreibern beispielsweise auch Dateisysteme, Line-Discipline-Module und andere Software. Die von Santa Cruz Operation gewählte Methode stellt somit Binärkompatibilität zu den existierenden Standards unter allen Umständen sicher.

Dennoch kann der Anwender auch Kernel-Programme jedem beliebigen Prozessor im System zuordnen. Die mit SCO Unix System V/386 gelieferten Einheitentreiber sind bereits dafür vorgesehen. In einer ausführlichen Dokumentation wird erläutert, wie Multithread-Kernel-Programme zu verfassen sind.

Durch das Hinzufügen der SCO Unix Multiprocessor Extension entstehen keinerlei Änderungen an Schnittstellen des Standardbetriebssystems SCO Unix System V/386. Applikationen für die Uniprozessor-Version laufen ohne Veränderungen; dies gilt sowohl für bereits installierte als auch für neue Anwendungen.

Auch die Schnittstellen innerhalb des Kernels bleiben unverändert. Das bedeutet, daß auch Kernel-Programme in der Multiprozessor-Umgebung ohne Änderungen zum Ablauf gebracht werden können.

Aufgrund der eng gekoppelten Prozessorarchitektur wird in einem Multiprozessor-System nur eine Kopie von SCO Unix abgearbeitet. Folglich benötigt der Anwender auch nur eine Lizenz von SCO Unix.

Binärkompatibilität sichergestellt

Der von Santa Cruz Operation eingeschlagene Weg zur Unterstützung von Multiprozessor-Rechnern sichert die vollständige Kompatibilität zu SCO Unix sowohl auf der Binärcode- als auch auf der Quellcodeebene.

Insbesondere sind große Anstrengungen unternommen worden, um auch in der Multiprozessor-Version den Standards zu entsprechen, für die SCO Unix System V/386 entwickelt wurde. Hierzu gehören die System V Interface Definition (SVID), Posix 1003.1, Fips 151.1 und NCSC TCSEC C2. Es gibt somit keine neuen speziellen Systemaufrufe oder Compilerbesonderheiten, die für das Multiprocessing zu beachten wären.

Strategie

Die SCO Unix Multiprocessor Extension wird als Softwarepaket geliefert, das zu einem bereits installierten SCO Unix System V/386 hinzugefügt wird. Jedes Paket erlaubt die Nutzung einer weiteren CPU im Multiprozessor-Rechner. Das Extension-Paket ersetzt die busunabhängigen Teile des Kernel Linkkit und erzeugt automatisch eine neue Binärversion des Betriebssystems.

Der zusätzliche Prozessor wird initialisiert und in die Multithread-Verarbeitung einbezogen. Mit jeder neuen Prozessorkarte wird – wie dargelegt – eine zusätzliche Erweiterung benötigt. Das erlaubt dem Anwender den stufenweisen Ausbau von Rechnersystemen sowohl hardware- als auch softwareseitig.

Unix-Systemcode

SCO hat darauf geachtet, daß im Zuge der Multiprozessor-Orientierung ausschließlich der Kernel des Betriebssystems SCO Unix System V/386 berührt wurde. Da am User-Modus des Betriebssystems nichts geändert wurde, konnten diesbezügliche Testläufe vollständig entfallen.

Der Betriebssystem-Kernel umfaßt rund 100 000 Codezeilen. Durch die Multiprocessor Extension werden nur etwa 8000 Zeilen modifiziert und nur rund 12 000 Zeilen hinzugefügt. Auch am Kernel waren also nur rund 20 Prozent des gesamten Codes zu testen.

Die Entwicklung der Extension wurde unter Beachtung hoher Sicherheitsstandards durchgeführt. Hierzu gehörte insbesondere die stufenweise Umstellung des Kernel auf Multithread-Fähigkeit. Diese Vorgehensweise war möglich, da nicht angepaßte, „normale" Kernel-Module in der SCO-Implementierung auf einem Standardprozessor (Basisprozessor) abgearbeitet werden. Daher konnten jeder neue Systemaufruf und jedes neue Kernel-Modul strengen Prüfungen unterzogen werden, bevor die jeweils nächste Stufe realisiert wurde.

Der Multiprozessor-Kernel

Den Schlüssel für die Multiprozessor-Fähigkeit bildet der Austausch ausgewählter Teile des normalen Unix-Kernels gegen neue Module. Bei der Installation der Extension werden ausschließlich diese Module ersetzt. Applikations- oder User-Modus-Code wird durch die Erweiterung nicht berührt. Modifiziert wird ausschließlich der System Control Kernel des Betriebssystems. Die Erweiterung hat somit auch keinen Einfluß auf Utilities, auf die Systemadministration oder auf den vom System belegten Plattenspeicherplatz. Der Betriebssystem-Kernel sorgt beim Systemstart dafür, daß jeder Prozessor auf den Kernel zugreift. Es gibt somit nur ein Code- und nur ein Datensegment im System, das von allen Prozessoren genutzt wird. Jeder Prozessor arbeitet dabei System- und Benutzer-Tasks ab und trägt somit zum Gesamtdurchsatz des Rechners bei. Das SCO Unix Multiprocessor Extension Linkkit verhält sich exakt wie das SCO Unix System V/386 Linkkit. Das bedeutet, daß beispielsweise Einheitentreiber in der gewohnten Weise ohne Änderungen installiert werden können.

Die Kompatibilität reicht sogar so weit, daß zum Beispiel Dateisysteme, Pseudotreiber und andere Kernel-Modus-Software unverändert einsetzbar sind. Dateisysteme, die zum Multiprozessor-Betriebssystem hinzugefügt werden, funktionieren ohne Änderungen. Das neue Dateisystem wird von der Standard-CPU des Multiprozessor-Rechnerns ausgeführt.

Abb.: Compaq



Die Besonderheit des Compaq Systempro ist der EISA-Bus

Das standardmäßig in SCO Unix implementierte Dateisystem ist hingegen für den Multithread-Betrieb ausgelegt. Es kann somit auf jedem beliebigen Prozessor des Rechners ausgeführt werden. Externe Schnittstellen für Systemaufrufe werden durch die Extension ebensowenig geändert wie interne Mechanismen für Systemaufrufe. Selbst nicht dokumentierte interne Unix-Aufrufe hat SCO unverändert gelassen, weil diese teilweise von versierten Programmierern trotz der fehlenden Dokumentation verwendet werden. Allerdings werden die bestehenden Systemaufrufe durch die Extension für die Multithread-Ausführung vorbereitet.

Sofern für das Multiprocessing zusätzliche Informationen für den Kernel benötigt werden (zum Beispiel die Angabe der aktuellen CPU in der u-page), wird eine neue Datenstruktur hinzugefügt. In keinem Fall wird eine bestehende Struktur geändert, da dies zu Inkompatibilitäten führen könnte.

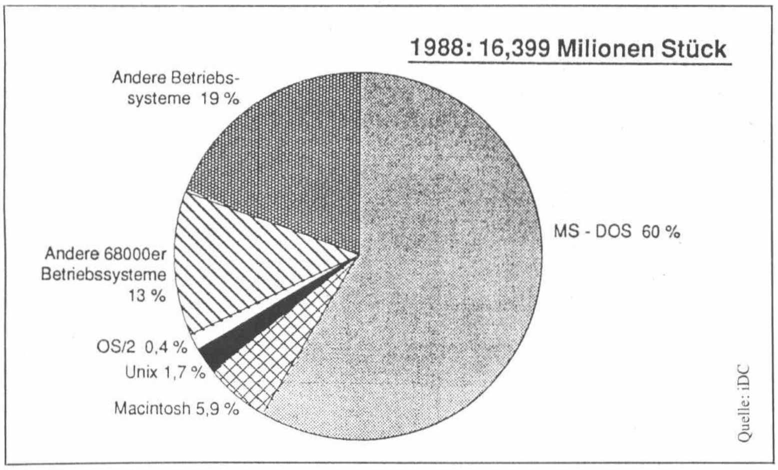
Es gibt im SCO Unix Kernel eine ganze Reihe von Unterroutinen, die nicht dokumentiert sind und von deren Verwendung durch externe Programmierer SCO ausdrücklich abrät. Dennoch werden diese Routinen, die aus Quell-Listings bekannt sind, von versierten Programmierern aus Effizienzgründen genutzt.

Das Unternehmen trägt dieser Tatsache Rechnung, indem auch diese internen Routinen durch die Extension unverändert blei-

ben. Die Kompatibilität ist somit auch für Applikationen sichergestellt, bei deren Entwicklung die von SCO vorgegebenen Programmierrichtlinien durchbrochen worden sind.

Hardwarearchitektur

Die von SCO mit der Standard-Extension derzeit unterstützten Multiprozessor-Rechner verfügen über eine Reihe von Gemeinsamkeiten, unterscheiden sich jedoch auch in wesentlichen Punkten, etwa hinsichtlich Performance und Konfiguration. Einer der Hauptvorteile der SCO-Softwarelösung besteht darin, daß sie bei aller Standardisierung den Hardwareherstellern einen breiten Spielraum bei der spezifischen Gestaltung ihrer Geräte einräumt. Obgleich die Entwicklung eines Multiprozessor-Rechners nicht einfach ist, werden durch die SCO-Software keine ungewöhnlichen Anforderungen an das Hardwarekonzept gestellt. Das bedeutet, daß die Extension als Standardsoftware künftig für eine Vielzahl unterschiedlicher Rechner zur Verfügung stehen kann.



PC-Betriebssysteme weltweit in Einheiten

Eine wichtige Voraussetzung für das Funktionieren der Erweiterung besteht darin, daß alle Prozessoren im Multiprozessor-Rechner auf den gesamten Speicher zugreifen können. Ein Byte an einer bestimmten Position muß für alle CPUs als dasselbe Byte an derselben Stelle verfügbar sein. Konvertierungsalgorithmen oder ähnliches können somit entfallen.

Die Lösung geht auch davon aus, daß kein einzelner Prozessor über einen gesonderten Speicherbereich verfügt. Ein Hardwarekonzept, das die Optimierung des Speicherzugriffs für alle CPUs gleichartig vorsieht, ist ohnehin am effizientesten; es wird von der Extension am besten unterstützt. Die Erweiterung kann bis zu 64 MByte Systemspeicher verwalten.

Die Instruktion xchg des Mikroprozessors Intel 80386 muß atomistisch über den gesamten Speicherbus wirken. Alle für das Multithreading notwendigen Lock-Algorithmen der Extension verwenden diese Instruktion, um Übertragungskollisionen durch die Prozessoren zu verhindern.

Jeder Prozessor muß seine eigenen Interrupts kontrollieren können. So wird beispielsweise der Basisprozessor (Standardprozessor) in der Regel den I/O-Bus steuern und dabei mit dem Programmable Interrupt Controller arbeiten. Jeder andere Prozessor hat je nach Hardwarekonzept ebenfalls Zugang zu diesem I/O-Bus, zu einem eigenen Bus oder nur auf den Systemtakt. Zudem sollte jeder Prozessor bei einem anderen Prozessor einen Interrupt auslösen können. Die Extension sieht vor, daß jeder Prozessor diese Interrupts maskieren und kontrollieren kann.

Bei einem Multiprozessor-Rechner werden erhöhte Anforderungen an das Speichersubsystem gestellt. Das hängt damit zusammen, daß bei einer Architektur für mehrere CPUs mehr Verarbeitungszyklen als üblicherweise gebraucht werden.

Um den Systemdurchsatz zu erhöhen, wird in den meisten Multiprozessor-Architekturen Cache-Speicher eingesetzt. Die Extension nutzt vorhandenen Cache-Speicher. Das gilt im übrigen auch für den Cache im Mikroprozessor 80486.

Ausführung des Multithreading

Für das Multiprocessing mußte der SCO Unix System V/386 Kernel um Locks erweitert werden. Locks sind kurze Datenstrukturen mit einem hochoptimierten Code für den Zugriff. Locks werden von allen Prozessoren erkannt; je nach Kontext wartet der Prozessor entweder bis zur Freigabe dieses Kernel-Teils, oder er arbeitet an einer anderen Aufgabe weiter. Das Ziel bei der Entwicklung der Extension war es, die Lock-Bereiche und die Lock-Zeiten im Kernel auf ein absolut notwendiges Minimum zu beschränken. Dadurch wird das Multithreading im Kernel besonders effizient.

Der Systemdurchsatz ist wesentlich höher, wenn Locks auf einzelne Ressourcen statt auf Kommandos gelegt werden. Es ist somit günstiger, eine Datei (Ressource) als die Ausführung einer write-Anweisung (Kommando) zu blockieren. Der Grund: Der Overhead ist bei einem Kommando durchweg umfangreicher als bei einer Ressource.

Die Theorien über Multiprocessing stellen durchweg fest, daß ein ausgewogener Mittelweg zwischen vielen Locks für kleine Codebereiche und wenigen Locks für umfangreiche Codebereiche das Optimum darstellt. SCO ist diesen Weg mit dem Resource-Locking gegangen. Die Locks werden gemäß der Nutzung der Ressourcen gesetzt, wobei die SCO Unix Multiprocessor Extension 25 Ressource-Klassen unter-

scheidet (Inodes, Run Queues, Sleep Queues, MMU Page Tables etc.).

Mit dem Zugriff auf Ressourcen und dem Einrichten von Zeigern auf Ressourcen werden Locks verschachtelt. In der Erweiterung wird dieser Tatsache durch einen Lock-Stapel Rechnung getragen. Hierbei erfolgt die Lock-Verwaltung durch einfaches Ablegen im Speicher und Zurückholen aus dem Speicher. Der Aufwand beschränkt sich auf ein Minimum.

An den Lock-Code werden hinsichtlich Verarbeitungsgeschwindigkeit höchste Anforderungen gestellt, da er im laufenden Betrieb ständig aufgerufen wird. Der Lock-Code in der SCO Unix Multiprocessor Extension wurde manuell optimiert; er verwendet die Instruktion xchg des Mikroprozessors 80386. Der Code greift lediglich auf zwei Bytes zu: ein Busy-Signal und eine Kennung des Eigentümers des Lock-Aufrufs.

Auf einem 386-Prozessor mit 16 MHz Taktfrequenz (die langsamste von der Extension unterstützte CPU) benötigt ein Lock nur 16,6 Mikrosekunden, ein Unlock 12,1 Mikrosekunden. Bei einem 486-Prozessor mit 25 MHz Taktgeschwindigkeit ergeben sich weniger als 4 Mikrosekunden für einen Lock und 3 Mikrosekunden für einen Unlock. Damit entsteht praktisch kein Overhead für den Kernel.

Interrups im Kernel

Die Extension ist hinsichtlich der Handhabung von Interrupts flexibel. Die Programmable Interrupt Controller werden für jeden Prozessor initialisiert. Für die Interrupts ist kein spezielles Handling notwendig – mit Ausnahme des Systemtaktes.

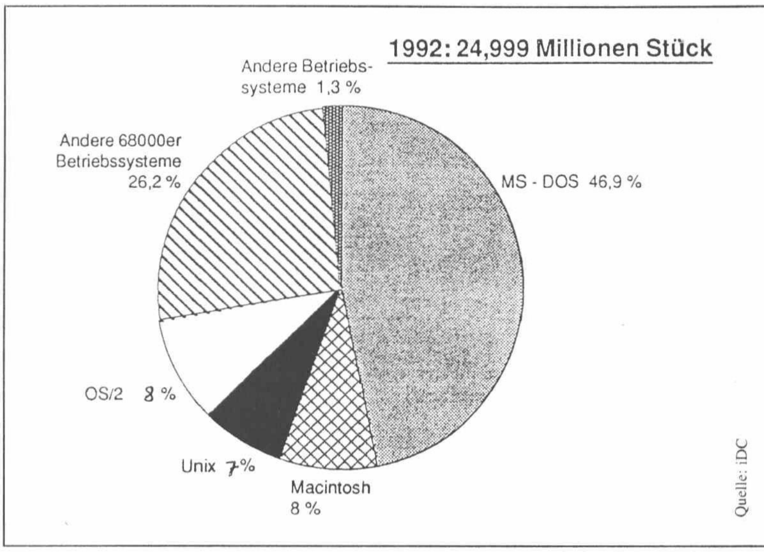
Das Betriebssystem Unix muß bei jedem Systemtakt eine Vielzahl von Operationen durchführen. In einem Rechner mit mehreren Prozessoren, die gleichzeitig arbeiten, würde es erhebliche Schwierigkeiten verursachen, wenn jede CPU von der Hardware gleichermaßen mit dem Systemtakt versorgt würde.

Daher erhält unter der Kontrolle der Extension nur ein einziger Prozessor den Systemtakt direkt von der Hardware. Die Extension verteilt den Takt anschließend optimal auf die anderen CPUs. Die Prozessoren erhalten den Systemtakt jeweils um einen Takt versetzt. Dadurch werden Lock-Kollisionen bei den standardmäßig notwendigen Interrupts von vornherein vermieden.

Einheitentreiber

Einheitentreibern stellen einen der schwierigsten Aspekte bei einem Multiprozessor-Betriebssystem dar. So ist die Multiprozessor-Hardware bezüglich des I/O-Buses im allgemeinen asymmetrisch. Das bedeutet, der Bus wird von nur einem Prozessor gesteuert, auf dem somit auch die Einheitentreiben laufen müssen. In einigen Dual-Processor-Architekturen können immerhin auch beide CPUs auf den Bus zugreifen.

Verfügt das Gerät über einen I/O-Standardbus, so erwartet der Anwender zu Recht, daß die I/O-Standardkarte ebenso wie der zugehörige Treiber unmittelbar funktionieren. Hinzu kommt, daß viele Treiber völlig anderen Aufgaben als der Bus-Unterstützung dienen, bei denen der Software oftmals kein Hardware-Äquivalent gegenübersteht (zum Beispiel für Warteschlangen oder Puffer). In diesen Fällen sind die Erwartungen der Anwender bezüglich Kompatibilität im allgemeinen noch höher.



Unix im Aufwind

Multithreading

Ein wesentliches Ziel bei der Entwicklung der Extension war es, Multithreading zu implementieren, ohne dadurch die Kompatibilität zur existierenden Treibersoftware zu gefährden. Zugleich waren Wege zu suchen, so daß neue Treiber für höchste Ansprüche an die Perfomance entwickelt werden können.

Die Lösung: Kernel-Module, die nicht speziell für den Multithreading-Betrieb vorgesehen und als solche gekennzeichnet sind, werden von einer Standard-CPU (Basisprozessor) bearbeitet. Das ist in der Regel der mit dem I/O-Bus verbundene Prozessor. Alle nicht modifizierten Treiber und Kernel-Module werden also auf dieser Basis-CPU abgearbeitet.

Das bedeutet, daß sämtliche Kernel-Modus-Module, die dem Extension-Kernel hinzugefügt werden, funktionieren. Da der Standardprozessor der einzige ist, der diese Module bearbeitet, kompensieren die anderen Prozessoren automatisch die Verarbeitungslast im Gesamtsystem.

Die bereits erläuterte Lock-on-Resources-Technik läßt sich auch für Treiber verwenden. Hierbei wird das Treiberprogramm nach einer sorgfältigen Analyse mit Locks für die vom Treiber beim Multithread-Betrieb benötigten Ressourcen versehen. Der neue Treiber wird der Extension hinzugefügt. Damit wird das Treiberprogramm ein integraler Bestandteil der SCO Unix Multiprocessor Extension. Die Standardtreiber von SCO Unix System V/386 sind durch die SCO Unix Multiprocessor Extension bereits für den Multithread-Betrieb modifiziert.

Dokumentation

Zusammen mit der Multiprocessor Extension wird eine separate Dokumentation für Systemprogrammierer geliefert. In dieser wird beschrieben, wie man Treiberprogramme für Multithreading vorbereiten kann, wenn dies etwa aus Performancegründen gewünscht wird. Nach der Modifizierung kann die Treibersoftware von einem beliebigen Prozessor innerhalb des Multiprozessor-Rechners vollständig abgearbeitet werden – bis auf die Ebene der I/O-Register, sofern die Hardware dies zuläßt.

Systemaufrufe

Einen wichtigen Faktor stellen Systemaufrufe dar, weil der Benutzer bei Systemaufrufen Zugriff auf die Kernel-Verarbeitung nimmt. Systemaufrufe sind die einzige Möglichkeit für ein Applikationsprogramm, Services des Unix-Kernels anzufordern. Hierbei wurde das Multithreading der Systemaufrufe so ausgelegt, daß ein Maximum an Serviceleistung durch jeden Prozessor innerhalb der der Applikation zugewiesenen Zeitspanne erbracht werden kann.

Der größte Teil der Services, die ein Anwendungsprogramm benötigt, wird durch einige wenige Systemaufrufe bereitgestelt. So nutzen die meisten Programme die Kommandos read und write intensiv für Ein- und Ausgaben bei Dateien, Netzwerken, Druckern und Terminals. Daher wurden diese Systemaufrufe als erstes für Multithreading vorgesehen. Analysen haben klar gezeigt, daß zwanzig Systemaufrufe etwa 80 Prozent der Verarbeitungszeit, die durch Systemaufrufe insgesamt beansprucht wird, verursachen. Diese Systemaufrufe wurden für den Multithread-Betrieb modifiziert. ■

Die Zukunft der Echtzeitverarbeitung

Stefan Klusmeier

Obwohl die parallele Datenverarbeitung noch in ihren Kinderschuhen steckt, stellt sie die Zukunft der Echtzeitverarbeitung dar. Die Frage, die viele Anwender sich stellen, ist: Sollen wir jetzt in Parallel Processing investieren oder abwarten?

Das Problem liegt darin, daß zwar nahezu monatlich neue Multiprozessor-Hardware-Plattformen vorgestellt werden, die verfügbare Menge an Software für diese Systeme jedoch äußerst gering ist. Dadurch fühlen sich viele neue Anwender von Multiprozessorsystemen von den Anbietern alleingelassen.

Gewöhnt an die Softwareflut, die während der letzten 30 Jahre für die traditionelle sequentielle Datenverarbeitung entwickelt wurde, fragen sich die Anwender von Multiprozessorsystemen, ob sie nun wieder Jahrzehnte warten müssen, bis die Software für Parallel Processing den Stand erreicht, den die Hardware schon heute bietet. Inzwischen versuchen sie, ihre sequentiellen Fortran-Programme an die neuen Plattformen anzupassen und zu optimieren, um sie unter diesen Bedingungen lauffähig zu machen. Dabei bietet der Markt für Anpassungs- und Tuning-Tools genausowenig wie der Markt der Anwendungssoftware für Parallel-Processing-Systeme.

E/SP schlägt die Brücke

Mit einem neuen Produkt von Concurrent Computer wird die Kluft zwischen dem „Was gibt es an Tools“ und dem „Was brauchen wir“ verringert. Das Environment for Sequential-to-Parallel-Processing, kurz E/SP, ist eine neue grafische Programmierumgebung für die Datenverarbeitung auf Multiprozessorsystemen. Dieses Produkt, das erste seiner Art auf dem Echtzeitmarkt, konvertiert schnell, effizient und kosteneffektiv sequentielle Programme in parallele Form. Es setzt den Anwender in die Lage, seine früheren Investitionen in sequentielle Fortran-Programme durch Transformation in leistungsstarke Parallelprogramme zu sichern.

Bisher war die Transformation und Optimierung von Programmen ein aufwendiger und daher auch teurer Prozeß. Der Programmierer mußte manuell die Stellen suchen, an denen das Programm durch Änderungen beschleunigt werden konnte. Man stelle sich einen Programmierer vor, dem die Aufgabe übertragen wurde, ein für ein Single-Prozessor-System geschriebenes Softwarepaket an die Anforderungen eines Sechs-Prozessor-Systems anzupassen.

Der Programmierer muß Zehntausende von Befehlszeilen daraufhin untersuchen, ob sie in ihrer internen Struktur ein gleichartiges Zeitverhalten besitzen. In regelmäßigen Abständen muß das Programm compiliert und ausgeführt werden, um festzustellen, ob eine Laufzeitbeschleunigung erreicht wurde und die Ergebnisse unverändert korrekt bleiben.

Dabei haben die besten Programmierer nach mehreren Monaten Arbeit keine Gewähr dafür, daß in dem konvertierten Programm alle möglichen Parallelitäten entdeckt worden sind und damit die volle Leistung erreicht wurde.

E/SP bietet einen Ausweg aus diesem Dilemma. Das Produkt vermeidet die aufreibende Arbeit und den möglichen Verlust von Investitionen in die Anwendungssoftware.

Visuelle Programmierung macht den Unterschied

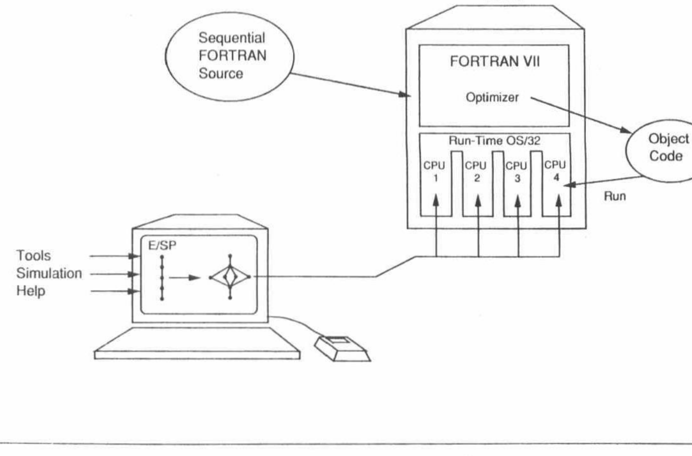
E/SP arbeitet mit visuellen Programmiertechniken. Nachdem ein sequentielles Programm in die E/SP-Datenbank eingeladen wurde, kann es in grafischer Form auf einem Workstation-Bildschirm dargestellt werden. Dabei kann festgestellt werden, welche Nebenläufigkeiten im Programm gefunden wurden.

Die einheitliche grafische Oberfläche von E/SP bietet dem Programmierer eine Alternative zum Lesen und Bearbeiten von Textzeile auf Textzeile. Das Window-Format ermöglicht dem Programmierer, in das Programm hineinzuschauen und über den Cursor oder die Maus Verbesserungen vor-



Der Autor Stefan Klusmeier schloß 1974 sein Hochschulstudium in Hannover als Diplom-Mathematiker ab. Heute ist er als Leiter Marketing bei der Concurrent Computer GmbH in Planegg/Martinsried tätig.

Schemabild von E/SP: Ein sequentielles Fortran-Programm wird optimiert und compiliert.



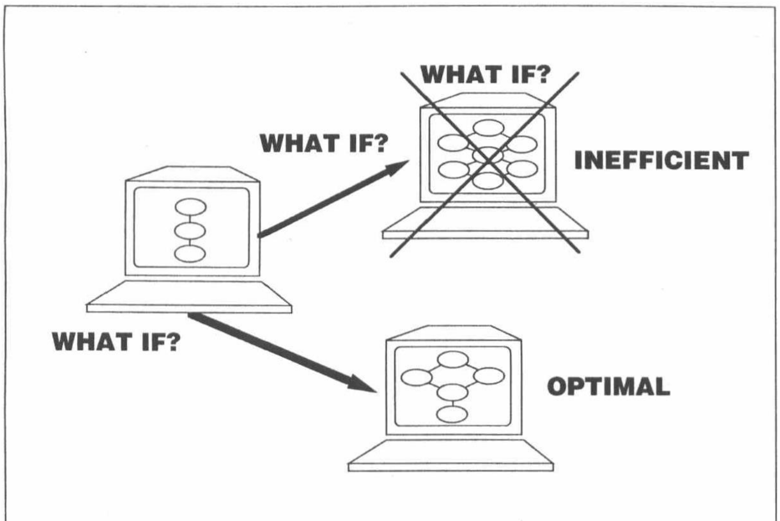
Abbildungen: Concurrent Computer

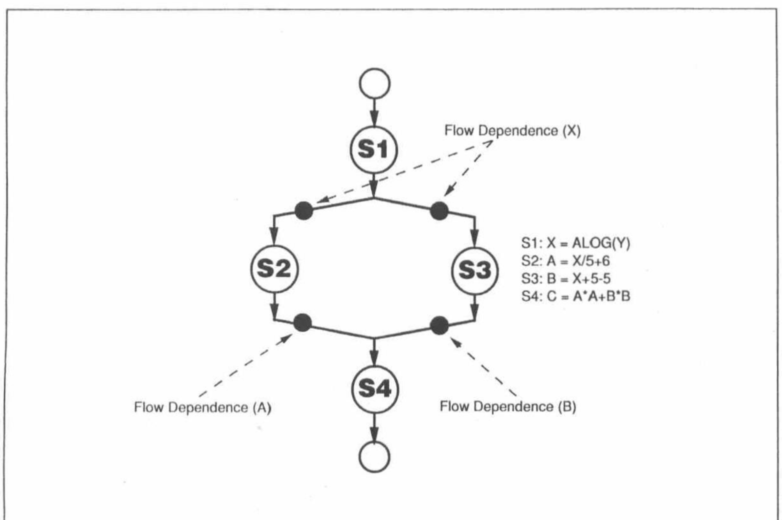
zunehmen. Pop-up-Menüs und Hilfefunktionen begleiten den Programmierer sicher bei jedem Schritt des Optimierungsprozesses. Zudem lassen sich durch Text-Windows Detailveränderungen im Quellcode vornehmen.

Die Simulation von E/SP ermöglicht das Tuning von Echtzeitanwendungen, um ein gefordertes Zeitverhalten zu realisieren.

Schleifen- und schleifenlose Parallelität

E/SP besitzt die Fähigkeit, parallele Strukturen zu definieren, die nicht in Programmschleifen liegen. Weniger fortgeschrittene Techniken, etwa automatisch parallelisierende Compiler, sind nur in der Lage, „Do"-Schleifen zu entdecken und sie aufzusplitten, damit sie parallel auf mehreren Prozessoren laufen können. E/SP dagegen kann Parallelitäten identifizieren, die nicht innerhalb von Schleifen liegen. Außerdem findet das Produkt unabhängige Sektionen heraus, die gleichartige Ausführungszeiten besitzen.





E/SP erkennt Parallelität außerhalb von Programmschleifen durch die Analyse von Datenabhängigkeiten und implementiert automatisch den parallelen Ablauf der Programmbereiche.

Ein gutes Beispiel ist der Whetstone-Benchmark, eine Serie kleiner sequentieller „Do"-Schleifen. Kleine „Do"-Schleifen über mehrere Prozessoren zu verteilen ergibt keinen Sinn, da der zusätzliche Overhead, der für die Synchronisation notwendig ist, die Ausführung deutlich verlangsamt.

E/SP sieht jede sequentielle „Do"-Schleifen-Struktur wie einen unabhängigen Programmteil an, der mit anderen unabhängigen Sektionen parallel verarbeitet werden kann. Das Resultat der Identifizierung von Programmteilen, die nicht unbedingt in Programmschleifen liegen, aber dennoch parallel bearbeitet werden können, ist eine optimale Leistungsausbeute. Während unter Verwendung von Tools früherer Generationen das Whetstone-Benchmark-Programm nicht parallelisiert werden konnte, erreicht E/SP 87 Prozent Steigerung auf einem Zwei-Prozessor-System.

Zudem wird das Potential parallel bearbeitbarer Sequenzen durch die automatisch durchgeführte interprozedurale Abhängigkeitsanalyse und die Entdeckung von internen, ineinander verschachtelten Parallelitäten weiter gesteigert. Sobald eine dieser inhärenten Parallelitäten entdeckt wird, wird diese Information auf dem Bildschirm der Workstation angezeigt. Durch die Manipulation grafischer Piktogramme, die verschiedene Programmstrukturen darstellen, kann dann der Programmierer weitere parallel bearbeitbare Sequenzen erzeugen.

Leistungssteigerung

Mit Granularität wird die Feinheit der Unterteilung eines Programms in kleinere Abschnitte bezeichnet, die parallel auf mehreren Prozessoren ausgeführt werden. Die Fähigkeit, Parallelität mit verschiedener Granularität anzubieten, ist wesentlich für Echtzeitapplikationen. Es ist notwendig, den Grad der Unterteilung gegenüber dem damit einhergehenden System-Overhead auszubalancieren, um eine effiziente Programmausführung zu gewährleisten.

E/SP bietet die programmierergesteuerte Parallelität in fast beliebiger Feinheit, indem mögliche Parallelitäten bis hinunter auf die Ebene des Statements dargestellt werden. Dabei ist der gewünschte Grad der Unterteilung von dem Programmierer frei wählbar.

So kann der Programmierer mit E/SP eine Reihe von Programmstrukturen und ihr Laufzeitverhalten simulieren, um die optimale Struktur herauszufinden. Sobald dieser Zustand erreicht ist, fügt das Tool die entsprechenden Anweisungen in den veränderten Sourcecode ein. Damit ist das optimierte Programmodell implementiert.

E/SP wurde für die Umsetzung von Programmen für die Multiprozessorsysteme der Rechnerfamilie 3200 von Concurrent Computer unter dem Echtzeit-Betriebssystem OS/32 und dem Fortran-VII-Compiler entwickelt. ■

Unix-Rechner im Vormarsch

Thomas Woywod

Parallel-Processing zeichnet sich immer stärker als einer der wichtigen Trends in der Informationstechnologie ab. Parallel-Processing ist eine zukunftsweisende Technologie, die hohe Leistung zu vergleichsweise geringen Preisen für viele Anwendungsfelder zur Verfügung stellt. Darüber hinaus ist diese Technologie geeignet, neue Anwendungsbereiche überhaupt möglich zu machen und zu erschließen, die bislang weder denkbar noch realisierbar waren. Die technologische Herausforderung liegt jedoch nicht in der Optimierung der Einzelkomponenten, sondern in deren Integration.

Parallel-Processing ist das Arbeiten mehrerer Prozessoren in einem einzigen System. Die Leistung solcher Computer mit paralleler Architektur reicht vom Mini- bis zum Großrechner, von rund 8 Mips bis 120 Mips (Millionen Instruktionen pro Sekunde), im Falle der Symmetrie-Serie von Sequent.

Parallel-Processing – Multiprocessing

Der Begriff „parallel" wird im Zusammenhang mit Computern in mehrfacher Bedeutung gebraucht. Der Begriff Parallel-Processing wird meistens für Parallel-Computing-Techniken gebraucht, die die Leistung eines Computers bei einer einzigen Anwendung beschleunigen.

Der Begriff Multiprocessing bezeichnet die Nutzung eines Mehrfach-CPU-Systems, die die Ausführung verschiedener unabhängiger Aufgaben beschleunigt.

Der Hauptunterschied zwischen diesen beiden Verarbeitungsarten ist, daß Programme für Multiprocessing nicht speziell durch den Programmierer strukturiert werden müssen, bevor sie von verschiedenen CPUs ausgeführt werden. Dies ist jedoch beim Parallel-Computing der Fall.

Das Prinzip des Parallel-Processing an sich ist nicht neu: Fast alle Computer nutzen bereits parallele Verarbeitung in Form von „intelligenten" Input/Output-Controllern, Floating-Point-, Grafik- und anderen dediziert arbeitenden Prozessoren.

Die verschiedenen Implementationen unterscheiden sich jedoch nicht nur in bezug auf Leistung, sondern auch auf Erweiterbarkeit und höhere Systemverfügbarkeit.

Tightly Coupled – Loosely Coupled

Bei einem eng gekoppelten System (tightly coupled) sind alle Prozessoren über einen High-Speed-Systembus mit einem gemeinsamen Hauptspeicher verbunden. Eine lose gekoppelte (loosely coupled) Konfiguration sieht für jede der CPUs ein eigenes Memory vor. Eine solche Konfigurierung macht es vergleichsweise schwieriger, unter den einzelnen CPUs Aufgaben zu verteilen. Die Zahl der Prozessoren ist allerdings auf mehrere tausend erweiterbar. Eng gekoppelte Systeme unterstützen weniger Prozessoren, echte Parallelverarbeitung ist dagegen einfacher realisierbar.

Symmetrie – Asymmetrie

Eine asymmetrische oder Master/Slave-Konfiguration hat einen „Master"-Prozessor, der Input/Output, Steuerung und andere Funktionen erfüllt, während „Slave"-Prozessoren die ihnen zugewiesenen Teile einer Aufgabe bearbeiten. Solche Systeme sind vergleichsweise einfach zu implementieren, finden bei steigender Arbeitslast schneller ihre Grenzen, da der Schnelligkeit eines „Master"-Prozessors Grenzen gesetzt sind.

In einem symmetrischen Parallel- oder Multiprozessorsystem arbeiten alle Prozessoren „gleichberechtigt" und parallel an den Aufgaben und sind in der Lage, Input/Output und andere Systemservicefunktionen auszuführen.

Aus diesem Grund kann einerseits eine größere Anzahl von Prozessoren genutzt werden, als bei Master/Slave-Konfigurationen der Fall ist. Außerdem können mehr Benutzer gleichzeitig bedient werden.

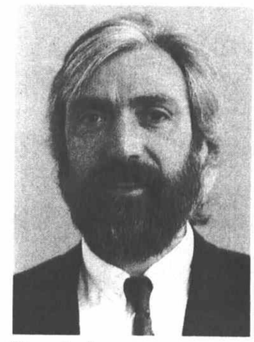
Statisch – Dynamisch

In einem statischen System wird ein Job oder Benutzer einem Prozessor zugewiesen, der den gesamten Job ausführt. Ein dynamisch gesteuertes System dagegen verteilt die Arbeitsbelastung gleichmäßig über das System. Jeder Prozessor arbeitet jeweils an der Aufgabe mit der höchsten Priorität, so daß kein Prozessor überlastet oder ohne Aufgabe ist.

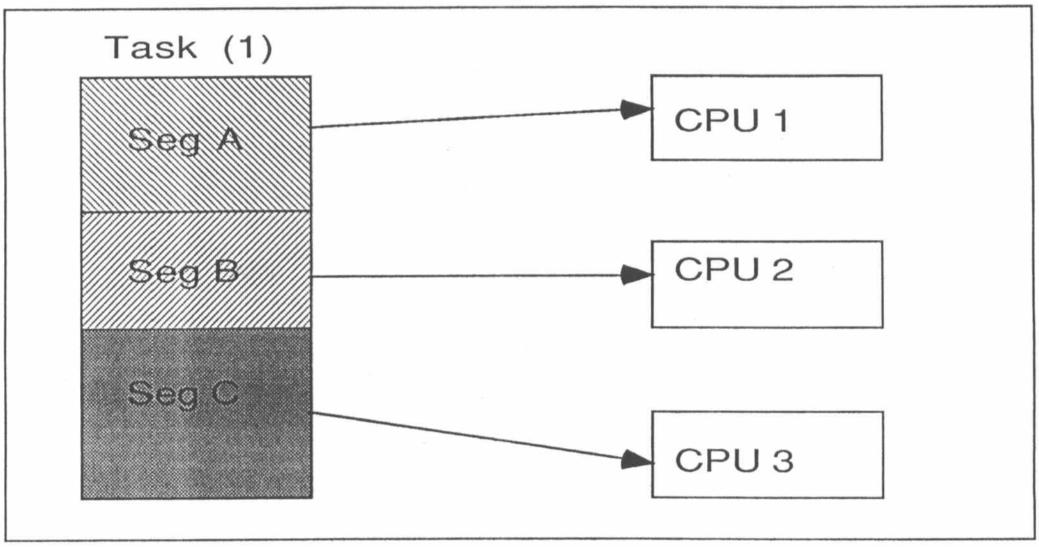
Anwendungsbereiche für Parallel-Computing

Die neue Technologie ist für alle Anwendungsgebiete besonders interessant, die viele aufteilbare Aufgaben enthalten, bei denen eine große Anzahl von Benutzern unterstützt werden soll und die hohe Leistung und Durchsatz, leichte Erweiterbarkeit sowie einen hohen Grad an Verfügbarkeit erfordern.

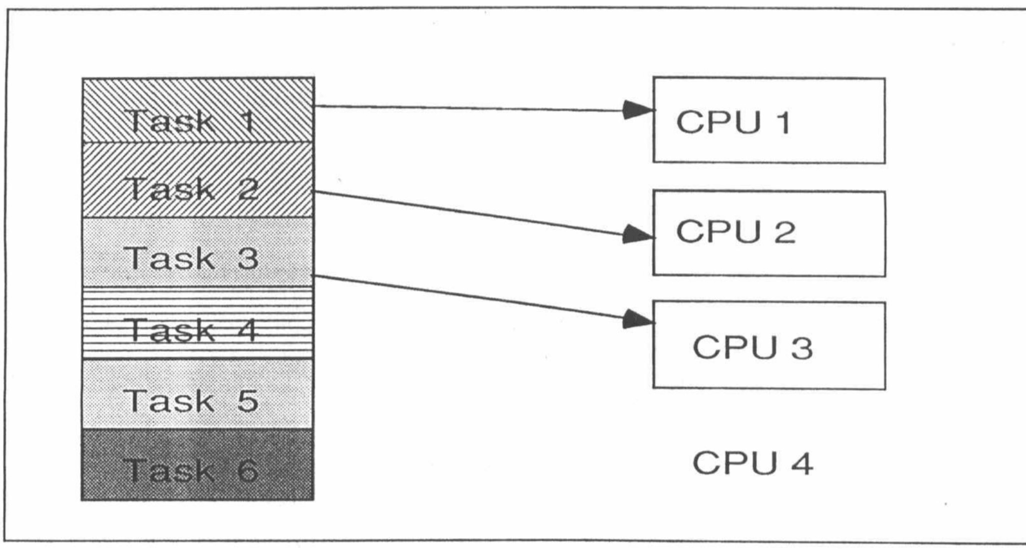
Im kommerziellen Bereich wird Parallel-Processing zunehmend interessanter: Gerade für Online-Transaktionsverarbei-



Der Autor Thomas Woywod ist Leiter Marketing Zentraleuropa bei der Sequent Computer Systems GmbH in München.



Parallel-Processing erfordert eine spezielle Strukturierung des Programms.



Multiprocessing erfordert keine spezielle Strukturierung der Software.

tung (OLTP) wie Flug- und Hotelreservierungssysteme, Logistiksysteme, Relationale Datenbanken und andere Aufgabenfelder bieten Parallelsysteme eine im Vergleich zu herkömmlichen Großrechnern kostengünstige Alternative.

Aber auch für den wissenschaftlichen Bereich ist Parallel-Processing interessant: Software-Entwicklung, Simulation oder Künstliche Intelligenz sind hier einige Anwendungsbeispiele.

Sequent-Systeme

Sequent entwickelt, produziert und vertreibt die Symmetry-Serie. Das sind symmetrische Parallel-Systeme (tightly-coupled), die die Programmabarbeitung dynamisch auf einen Anzahl von zwei bis zu 30 Prozessoren verteilen. Die Symmetry-Serie nutzt Intel-80386-Mikroprozessoren als CPUs und deckt so eine Leistungsbandbreite von acht bis zu 120 Mips unter einem einzigen auf Unix basierenden Betriebssystem ab.

Die Symmetry S27 kann mit zwei bis zu zehn Prozessoren konfiguriert werden und unterstützt bis zu 96 Benutzer.

Die Symmetrie S81 ist mit maximal 30 CPUs konfigurierbar und kann bis zu tausend Anwender unterstützen.

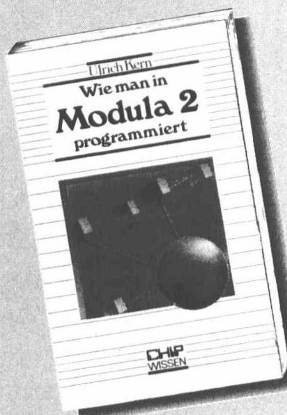
Die Preise für Systeme der Symmetry-Serie bewegen sich zwischen 300 000 und mehr als drei Millionen Mark.

Das Dynix-Betriebssystem von Sequent ist eine um Parallelität erweiterte Version von Unix. Dynix unterstützt parallelen Mehrfachprozessor-Betrieb und ermöglicht einen quasi-linearen Leistungsanstieg, wenn weitere Prozessoren hinzugefügt werden. Dynix ist sowohl zu Unix System V als auch Unix BSD 4.2 kompatibel; Programme, die für eines der beiden Unix-Versionen geschrieben wurden, können auch mit Dynix betrieben werden.

Schon mehr als 200 kommerzielle Anwendungen sind für Sequent-Parallelrechner portiert. Auch technisch-wissenschaftliche Anwendungen sind verfügbar. ■

CHIP WISSEN

Die kompetente Buchreihe rund um den PC

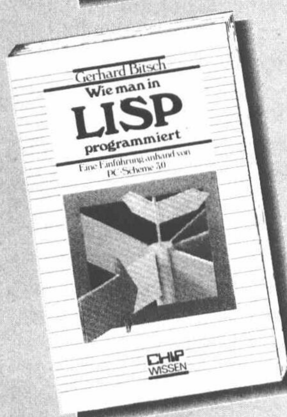


Ulrich Kern

Wie man in Modula 2 programmiert

256 Seiten, zahlr. Bilder, Hardcover
40,– DM/ISBN 3-8023-**0207**-9

In Modula 2 geschriebene Programme sind in hohem Maß portierbar, d.h. mit wenig Aufwand auf verschiedenen Rechnern einsetzbar. Modula-2-Compiler gibt es für nahezu jeden Computer, vom PC bis zum Großrechner.
Das Buch enthält einen vollständigen Einführungskurs in das Programmieren mit Modula-2.
Die Sprachkonzepte werden anhand zahlreicher Beispiele dargestellt. Übungen und deren exemplarische Lösungen vertiefen den behandelten Stoff. Der Leser lernt den Umgang mit Algorithmen und Datenstrukturen, von einfachen Anweisungen bis hin zu Rekursionen und Zeigertypen.



Gerhard Bitsch

Wie man in LISP programmiert

228 Seiten, 37 Bilder, Hardcover
45,– DM/ISBN 3-8023-**0182**-X

LISP ist eine der ältesten und wichtigsten Programmiersprachen der Künstlichen Intelligenz. Mit der Verfügbarkeit leistungsfähiger Versionen für Mikrocomputer stellt sie aufgrund ihrer einfachen Grammatik eine Alternative für den Programmiereinstieg dar.
Dieses Buch ist eine Einführung in den für IBM-Rechner entwickelten Dialekt PC-Scheme 3.0. Anhand von alltäglichen, einfachen Beispielen wird dem Einsteiger die Programmierung vermittelt und das Interesse geweckt, sich mit diesen Grundlagen auch an größere Projekte der Künstlichen Intelligenz zu wagen. Zahlreiche Übungsaufgaben mit Lösungen überprüfen und vertiefen die Kenntnisse.
Eine Diskette mit den im Buch enthaltenen Programmbeispielen ist zum Preis von 38,– DM erhältlich.

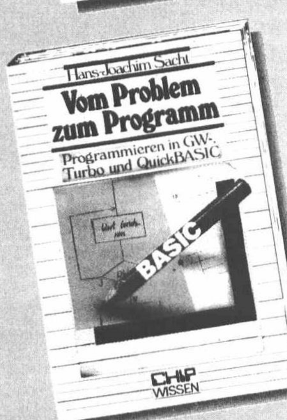


Hans-Joachim Sacht

Programmieren mit QuickBASIC 4.0 und 4.5

210 Seiten, 66 Bilder, Harteinband
mit 5,25-Zoll-Diskette
48,– DM/ISBN 3-8023-**0245**-1

Durch die neuen Microsoft-Compiler QuickBASIC 4.0 und 4.5 können so komfortable und gutstrukturierte Programme entwickelt werden, wie es sonst nur mit einem leistungsfähigen Interpreter möglich war.
Das Buch bietet die vielfältigen Funktionen in konzentrierter Form und geht vor allem auf die Handhabung und Sprachelemente dieses Compilers ein. Programmierern, die bisher nur mit einem BASIC-Interpreter gearbeitet haben, wird erklärt, wie sie schnell auf QuickBASIC umsteigen und so die Vorteile der Kompilierung ausnutzen können. Eine Diskette mit den im Buch behandelten Beispielen liegt bei.



Hans-Joachim Sacht

Vom Problem zum Programm

Programmieren in GW-BASIC, Turbo BASIC und QuickBASIC
464 Seiten, 183 Bilder
4., völlig überarbeitete Auflage 1988
48,– DM/ISBN 3-8023-**0214**-1

Das von Microsoft entwickelte GW-BASIC bzw. BASICA gilt inzwischen bei 16-bit-Computern als Standard und für 32-bit-Rechner als Grundlage des Sprachumfangs auch anderer neuer Dialekte.
Die Neukonzeption gliedert sich in 3 Teile:

1. Teil erklärt Grundlagen der BASIC-Programmierung unter Einbeziehung der wichtigsten MS-DOS-Kommandos, Compiler.
2. Teil bringt zahlreiche Programmroutinen für breite Anwendungen.
3. Teil zeigt, wie ein Programm Schritt für Schritt entsteht.

Eine Diskette mit den im Buch behandelten Beispielen ist erhältlich.

VOGEL

Haben Sie schon den neuen „CHIP- Katalog“? Bestellen Sie gleich!

Vogel Buchverlag
Postfach 67 40 · 8700 Würzburg 1

Neue Supercomputer-Generation

Norbert Vorstädt



Der Autor Norbert Vorstädt
ist als Vertriebsleiter bei der GEI-Rechner-Systeme in Aachen tätig.

Nahezu jeder Zweig wissenschaftlicher und technischer Arbeit ist heute von leistungsfähigen Rechnern abhängig. Einige Disziplinen sind sogar auf die schnellsten realisierbaren Rechner geradezu angewiesen. Für jene, die sich einen solchen Supercomputer leisten können, sind schnellste Elektronik und kürzeste Zykluszeiten, wie etwa bei einer Cray, der beste Weg zur Lösung numerisch intensiver Aufgaben. Wegen ihres hohen Preises bleiben solche Rechner für die Mehrzahl der Anwender außer Reichweite.

Vor wenigen Jahren tat sich eine Alternative in Form sogenannter „Mini-Supercomputer" auf, die eine gehörige Portion an Supercomputer-Leistung innerhalb des Budgetrahmens einer breiten Benutzergemeinde versprachen. Um eine hohe Rechnerleistung zu einem Bruchteil Cray-artiger Preise anbieten zu können, verknüpften solche Systeme konventionelle, relativ preiswerte Hardware mit bestimmten Formen der Parallelverarbeitung. Obwohl die Nachfrage nach Mini-Supercomputern einen hohen Bedarf an noch besseren Lösungen für rechenintensive Anwendungen belegt, setzen die meisten dieser Rechner „traditionelle" Parallelarchitekturen ein, die einen Großteil der durch Parallelisierung möglichen Leistungsgewinne unausgeschöpft lassen. Eine neue Rechner-Architektur namens „Very Long Instruction Word" (VLIW) ermöglicht die Nutzung dieses Potentials.

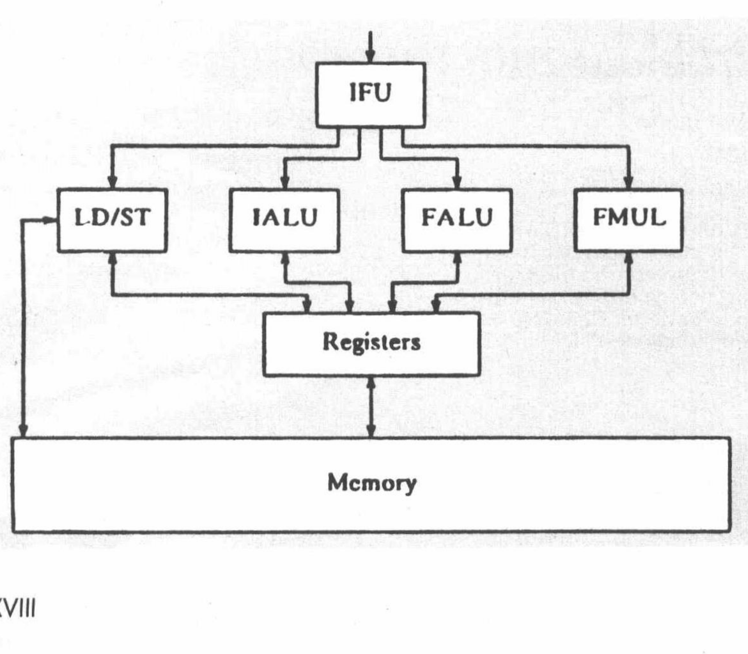
Noch vor kurzem konnte man Mini-Supercomputer danach kategorisieren, ob sie Vektor- oder Multiprozessortechniken einsetzten. Während Vektorrechner ihre Leistung dadurch erzielen, daß sie viele gleichartige Operationen auf einmal ausführen, arbeiten Multiprozessoren mit der Methode, mehrere voneinander unabhängige Codesequenzen gleichzeitig zu bearbeiten. Beide Arten von Rechnern setzen entsprechend passende Strukturen im Anwendungsprogramm voraus. Und das wiederum hat für den Benutzer eine Reihe von Konsequenzen, die wir näher betrachten wollen.

Vektorrechner erzielen ihre Leistung dadurch, daß sie einzelne Operationen auf eine Gruppe von Elementen anwenden. Statt also hundert Additionen hintereinander auszuführen – wie bei einem herkömmlichen Minicomputer –, stellen sie alle Operanden bereit und führen die Additionen in einer einzigen Instruktion aus. Ein Do-Loop ist ein Beispiel für eine Programmstruktur, auf die sich dieses Prinzip anwenden läßt und die im Programm vorhanden sein muß, damit ein Vektorrechner „auf Touren" kommt. Das Auffinden solcher Programmstrukturen, „Vektorisierung" genannt, kann teilweise automatisch vom Compiler durchgeführt werden; zum Teil muß der Programmierer bei der Vektorisierung eingreifen. Manchmal muß er sogar die Arbeitsweise des Programms völlig umgestalten, um eine Vektorisierung zu ermöglichen.

Würde ein Programm nur aus vektorisierbaren Strukturen bestehen, ließe sich mit Hilfe von Vektorrechnern eine enorme Leistungssteigerung erzielen. Doch für viele Anwendungsprogramme liefert die Vektoreinheit nur dürftige Gewinne. Das liegt daran, daß die meisten Programme aus einer Mischung von vektorisierbaren und nichtvektorisierbarem Code bestehen. Diese Mischung ist sehr anwendungsabhängig. Ist sie zum Beispiel 50/50, so kann selbst eine unendlich schnelle Vektoreinheit die Geschwindigkeit nur verdoppeln. Wirklich große Beschleunigungsfaktoren liefert die Vektortechnik also nur dann, wenn ein sehr hoher Prozentsatz des Codes vektorisierbar ist.

Bei den Multiprozessor-Maschinen liegt eine ähnliche Problematik vor. Das Auffinden von parallelisierbaren Programmteilen funktioniert nur zum Teil automatisch. Manchmal muß ein Programm drastisch verändert werden, um die Hardware des Multiprozessors effizient zu nutzen. Und letztlich erzielt man nur bei solchen Anwendungen eine angemessene Leistung, die zu einem sehr hohen Anteil parallelisierbar sind. Die meisten Anwender, insbeson-

Klassische sequentielle Architektur



dere diejenigen, die ein breites Spektrum von CAE-Programmen verwenden, suchen jedoch nach Rechnern, die nicht nur schneller sind, sondern deren Leistung weniger von der Art der Anwendung abhängt.

Die Trace-Rechnerfamilie bietet jetzt eine solche anwendungsunabhängige, hohe Rechenleistung. Sie nutzt die gleichzeitige Ausführung von einzelnen, unkorrelierten Maschinenbefehlen – eine Technik namens Overlapped Execution –, um zu einem hohen Grad an Parallelität zu gelangen. Die Präsenz dieser feinkörnigen Parallelität ist in keiner Weise von Art oder Struktur eines Anwendungsprogramms abhängig. Ein hoher Leistungsgewinn wird bei dieser Parallelisierungsmethode in allen Routinen aller Anwendungsprogramme erzielt. Aus diesem Grunde liefern die Systeme höhere Leistung bei einem breiteren Anwendungsspektrum als Systeme, die traditionelle Methoden der Parallelisierung verwenden.

Die Planung, welche Operationen gleichzeitig ausgeführt werden sollen, erfolgt vollkommen automatisch und erfordert nicht die Mitarbeit des Programmierers. Es profitieren also auch solche Programme von der hohen Rechenleistung, die seltener gebraucht werden und daher nicht unter großem Zeitaufwand optimiert werden können. Außerdem ist die Parallelisierungsmethode für den Benutzer völlig transparent, da sie unterhalb der Programmiersprachenebene erfolgt.

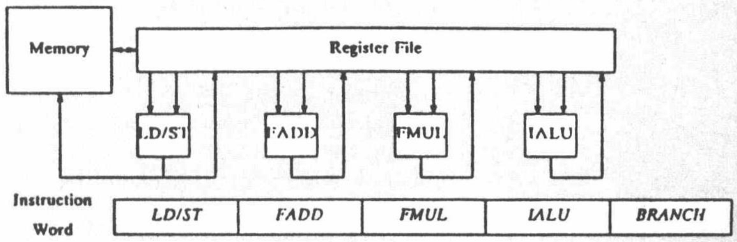
Diese Vorteile erzielt die Trace mittels zweier grundsätzlich neuer Techniken: ihrer VLIW-Architektur und einer Compilertechnik mit der Bezeichnung „Trace Scheduling".

Klassische sequentielle CPU

Der erste Schritt zum Verständnis der VLIW-Architektur besteht darin, sich vor Augen zu halten, wie eine klassische sequentielle CPU arbeitet. Vereinfacht dargestellt liest und decodiert die Instruction Fetch Unit (IFU) eines solchen Rechners eine Instruktion und bringt diese in einer der Funktionseinheiten zur Ausführung, die dann zum Beispiel ein Memory-Wort lesen oder eine Integer- oder Floating-Point-Multiplikation durchführen. Eine gebräuchliche Methode zur Leistungssteigerung ist das Pipelining von Instruction Fetch-, Decodier- und Ausführungsphase. Doch selbst dann arbeitet bei solchen Maschinen stets nur eine der vielen Funktionseinheiten.

Klassische CPU mit Overlapped Execution

CPU-Designern fiel diese Schwachstelle auf, und sie begannen in den frühen 60er Jahren, Hardware zu bauen, die es bei diesen sequentiellen CPUs ermöglichte, mehrere Funktionseinheiten zeitlich überlappend zu betreiben. Diese Hardware interpretierte eine normale sequentielle Instruktionsfolge, fand heraus, ob aufeinanderfolgende Instruktionen unabhängig voneinander bearbeitet werden konnten, und startete sie parallel, falls möglich.



VLIW-Architektur

Der Vorteil dieser Technik ist der Leistungsgewinn durch Parallelisierung, der universell wirksam und für den Programmierer transparent ist. Der Nachteil ist der immense Aufwand an Hardware, der dazu führt, daß diese Technik nur in Systemen der oberen Preisklasse eingesetzt werden kann. Außerdem haben Untersuchungen gezeigt, daß der maximal erreichbare Leistungsgewinn durch diese Technik etwa bei einem Faktor zwei liegt. Dies hat zwei Ursachen: Bei der Ausführung muß die Daten-Abhängigkeit (Data Precedence) streng berücksichtigt werden. Das Parallel-Dispatching muß für bedingte Sprünge unterbrochen werden.

Grenzen des Parallelismus

Betrachten wir diese limitierenden Faktoren einmal im Detail. Datenabhängigkeit ist recht einfach zu beschreiben. Es muß stets sichergestellt sein, daß zum Startzeitpunkt einer Instruktion ihre Operanden vorliegen und alle dazu nötigen Instruktionen bereits abgeschlossen sind. Beispiele:

```
A = (B + C) * (D + E)
F = (G * A) + (H * I)

A(I) = (B + C) * (D + E)
F = (G * A(J) + (H * I)
```

Im ersten Beispiel benötigt man zur Berechnung von F den Wert von A. Das zweite Beispiel zeigt einen ähnlichen Fall, wobei der Array-Charakter von A die Erkennung eines möglichen Konflikts erschwert.

Die Problematik der bedingten Sprünge ist ebenfalls leicht einsehbar. In der Sequenz

```
A = (B + C) * (D + E)
IF (A .GT. 1000.0) THEN
   F = (G * H) + (X * Y)
ENDIF
```

kann die Zuweisung an F nicht eher durchgeführt werden, bis man den Wert von A kennt und weiß, ob A größer ist als 1000.0.

Simulationen haben gezeigt, daß die bedingten Sprünge das größere Problem darstellen. Man war in der Lage, eine Maschine zu simulieren, die die Datenabhängigkeit strikt berücksichtigt, jedoch bei jedem bedingten Sprung beide Richtungen parallel verfolgt. Diese Maschine erzielte sehr hohe Leistungsgewinne durch die Overlapped Execution, die in der Größenordnung von zehn bis 50 lagen. Doch wäre die Realisierung einer solchen Maschine unmöglich gewesen, da sie 32 000 Prozessoren und andere unrealistische Dinge erfordert hätte.

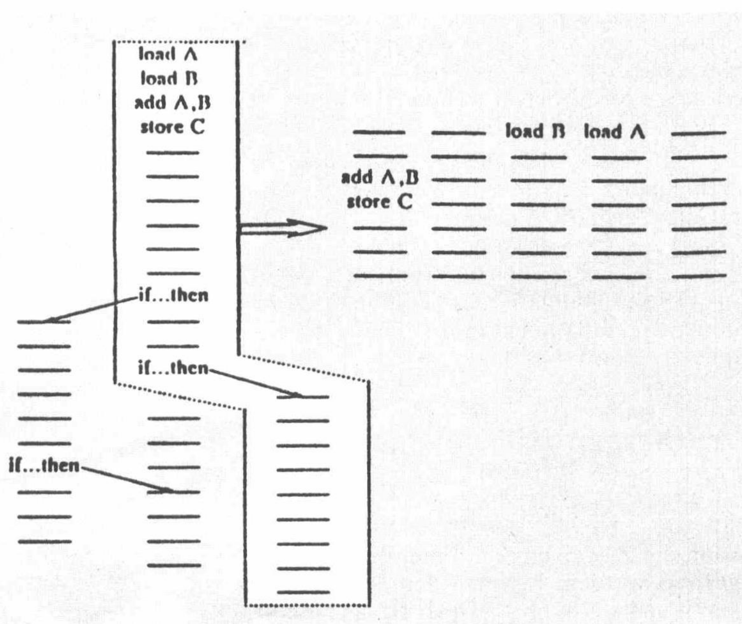
Übrigens hat diese Hürde zur Entwicklung von Vektorrechnern geführt, die eine spezielle Form der Instruktionsüberlappung verwenden, die unter bestimmten Umständen sehr gut funktioniert, jedoch nicht so allgemein anwendbar ist wie die Technik, die im folgenden beschrieben wird.

Die VLIW-Architektur stellt eine Form von sehr weiträumiger Overlapped Execution dar und geht diese Hürde der bedingten Sprünge direkt an.

Very Long Instruction Word (VLIW)-Architektur

Die VLIW-Architektur wurde speziell als Zielmaschine für Trace Scheduling Compiler entworfen. Bei einer VLIW-Maschine enthält jede Instruktion eine große Zahl von individuellen Operationen, die vom Compiler zur parallelen Ausführung bestimmt wurden. Sie gehört zur Klasse der Single-Prozessor-Rechner und hat nur einen Program Counter. Sie ist nicht mit einem Multiprozessor zu verwechseln. Der CPU-Kern ist der gleiche wie im vorigen Beispiel, jedoch liegt die gesamte Steuerung hier nicht in der Hardware, sondern im Compiler.

Trace Scheduling



Jede Instruktion enthält viele Operationen, und jede Operation wird von einer Art RISC-Prozessor ausgeführt. Die Maschine ist nicht mikroprogrammiert. Jede Instruktion wird in einem einzigen Zyklus ausgeführt. Benötigt eine Operation mehrere Zyklen, so wird sie in einer Pipeline bearbeitet, deren Steuerung voll in den Händen des Compilers liegt. In erster Näherung kennt und verwaltet der Compiler sämtliche Hardware-Ressourcen der CPU und des Memory-Systems. Die Beachtung der Datenabhängigkeit wird voll durch den Compiler übernommen. Bei der Codegenerierung findet ein Verdichtungsprozeß auf die langen Instruktionsworte statt, wobei gleichzeitig die zeitliche Planung und mögliche Verschiebung der einzelnen Operationen über die Grenzen von bedingten Sprüngen hinweg erfolgt.

Soweit die allgemeine Beschreibung der VLIW-Architektur. Eine Darstellung der Maschinen, die die amerikanische Firma Multiflow auf der Basis dieser Architektur entwickelt hat und die in Deutschland durch GEI Rechnersysteme (Aachen) vertrieben werden, folgt im Anschluß an die Betrachtung der Trace Scheduling Compiler, die ja der eigentliche Ausgangspunkt dieser Entwicklung waren.

Code-Verdichtung auf einem VLIW-Rechner

Das elementare Code-Generierungsproblem für eine VLIW-Maschine besteht aus der Analyse einer sequentiellen Operationsfolge, der Ermittlung der Datenabhängigkeiten und der Abbildung dieser Operationsfolge auf eine möglichst kleine Anzahl von VLIWs.

Dabei wird die lange Folge von Operationen auf der linken Seite in die breiten Instruktionswörter auf der rechten Seite umgeformt. Die Ausführungszeit des so generierten Codes entspricht der Länge des kritischen Pfades im Operations-Graphen. Alle anderen Operationen, die jetzt parallel zu denen des kritischen Pfades ablaufen, kosten keine zusätzliche Zeit mehr. Die Praxis zeigt, daß man die Anzahl der zu verdichtenden Operationen verdoppeln kann und – vorausgesetzt, man hat genügend Funktionseinheiten zur Verfügung – dabei auf nur wenig mehr Instruktionswörter kommt. Dadurch wächst die Effizienz des Compilers immens mit länger werdender Operationsfolge. Doch unglücklicherweise findet man keine langen ununterbrochenen Operationsfolgen in normalen Programmen, da etwa alle acht bis zehn Operationen ein bedingter Sprung auftritt.

Entsprechend dieser Beobachtung muß es das Entwicklungsziel sein, die bedingten Sprunggrenzen zu überwinden und Operationen über diese Grenzen hinweg verschie-

ben zu können. Und genau das ermöglicht die weiterentwickelte Compilertechnik des Trace Scheduling.

Trace Scheduling

Die Grundidee beim Trace Scheduling besteht darin, zur Compile-Zeit den Compiler einen langen Pfad von Operationen bestimmen zu lassen, der über viele bedingte Sprünge hinweggeht und in der Regel mehrere hundert Operationen umfaßt. Dieser Pfad wird entweder auf heuristischem Wege vom Compiler ausgewählt oder durch einen Probelauf des Programms ermittelt, immer mit dem Ziel, den Pfad mit der größten Ausführungshäufigkeit zu finden. Dieser Pfad wird dann vom Codegenerator als Einheit aufgefaßt und verdichtet, wobei durch zahlreiche Umplazierungen von Operationen eine Folge von dichtgefüllten VLIW-Instruktionen entsteht.

Kompensationscode

Solange bei der Ausführung der Programmfluß auf dem soeben verdichteten Pfad entlangführt, läuft unser Programm sehr schnell und vollkommen korrekt, denn wir haben bei allen Umplazierungen stets die Datenabhängigkeit beachtet. Nächstes Ziel ist es nun, dafür zu sorgen, daß der erzeugte Code überall korrekt läuft, unabhängig davon, welchen Weg die bedingten Sprünge einschlagen und insbesondere auch dann, wenn sie den gerade verdichteten Pfad verlassen.

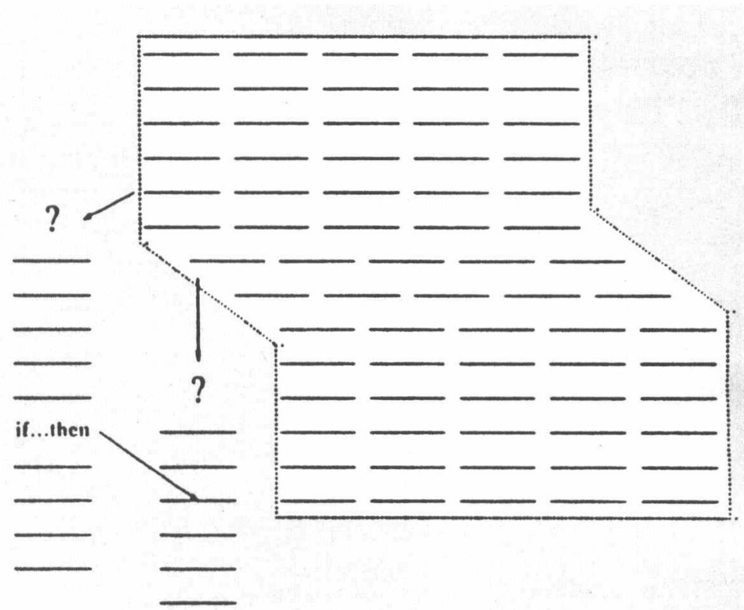
Um das sicherzustellen, erzeugt der Compiler an den Zielorten der bedingten Sprünge, die den verdichteten Pfad verlassen, sogenannten Kompensationscode. Dieser Kompensationscode gleicht den Unterschied zwischen dem ursprünglich erwarteten Programmstatus und dem aktuellen Programmstatus, der ja durch die Umplazierung von Operationen beeinträchtigt ist, wieder aus. Der Compiler ist dahingehend optimiert, diesen Kompensationscode möglichst kleinzuhalten.

So wie wir den ersten Pfad (Trace) verdichtet haben, fahren wir jetzt fort, bestimmen vom verbleibenden Code wiederum den Pfad mit der größten Ausführungshäufigkeit und verdichten ihn. Und wir iterieren nach diesem Schema solange, bis das gesamte Programm zu VLIW-Instruktionen verdichtet worden ist. Dabei wird übrigens der Kompensationscode zusammen mit den anderen Operationen eines Pfades verdichtet und wird somit parallel zu den „echten" Programmoperationen ausgeführt.

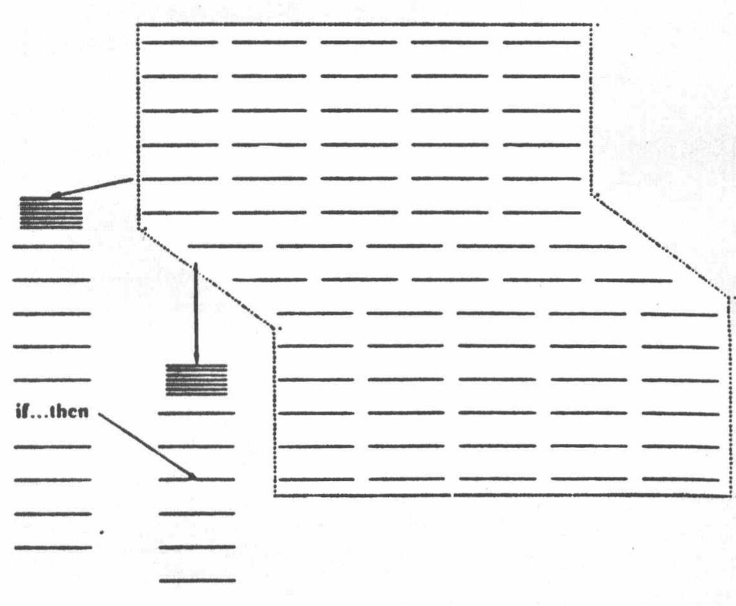
Die GEI Trace 7/200

Die Basis dieser neuen Rechnertechnologie wird also durch die Trace Scheduling Compiler und die VLIW-Architektur gebildet. Nun kann man Rechner nach dem VLIW-Konzept in weiten Bereichen dimensionieren, das heißt, man kann die Breite des Instruktionswortes vergrößern, eine entsprechende Zahl von Funktionseinheiten hinzufügen und braucht nur den Compiler von den hinzugekommenen Hardwareressourcen in Kenntnis zu setzen.

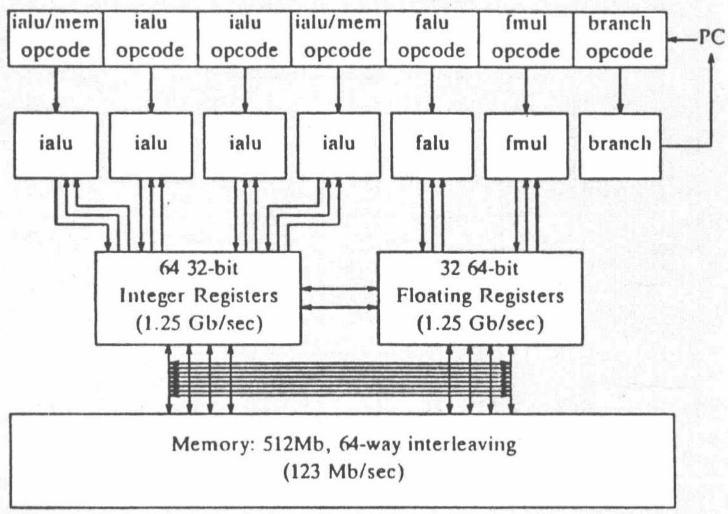
Die kleinste CPU in der Trace-Familie ist die Trace 7/200. Sie ist ein Single-Prozessor (mit nur einem Programm-Counter), und dieser hat ein 256 bit breites Instruktionswort, also ganze 32 Byte. Jede Instruktion führt sieben Operationen aus, davon vier Integer-Operationen, zwei Floating-Point-Operationen und einen Branch. Zwei von den vier Interger-Operationen können Load- und Store-Operationen sein. In jedem Maschinenzyklus werden eine Instruktion und somit sieben Operationen ausgeführt.



Erster Pfad verdichtet



Kompensationscode



GEI-Trace-7/200-Architektur

Memory-Referenzen und Floating-Point-Operationen werden in Pipelines ausgeführt. Diese Maschine führt 53 Millionen Operationen pro Sekunde aus, davon 30 Millionen Single Precision oder 15 Millionen Double-Precision-Floating-Point-Operationen.

Diese CPU besteht aus nur drei Boards und ist auf der Basis von CMOS Gate Arrays und TTL-Logik realisiert. Eines der Boards ist für die Integer-Operationen, ein weiteres für die Floating-Point-Arithmetik zuständig, und das dritte Board enthält alle Einheiten, die genau einmal benötigt werden, wie den Programm-Counter, Arbitration-Logik und die Interrupt-Hardware. Die Maschine kann im selben Schrank zu jedem der größeren Rechner der Trace-Familie ausgebaut werden.

Die größeren Mitglieder der Trace-Familie sind die 14/200 und die 28/200. Bei der 14/200 enthält jede Instruktion 14 Operationen, bei der 28/200 entsprechend 28 Operationen. Die 28/200 hat also die vierfache Breite einer 7/200, und damit vervierfachen sich auch alle Peak-Leistungsdaten der Maschine. Das Instruktionswort hat eine Breite von 1024 bit.

Unix auf diesen Maschinen

VLIWs wurden ursprünglich als Back-End-Prozessoren – auch unter der Bezeichnung Attached Processor bekannt – konzipiert. Die Rechner waren jedoch von Anfang an als Zielmaschinen für die Trace Scheduling Compiler geplant, und es erschien möglich, das 4.3 BSD Unix darauf zu portieren. Und genau das gelang.

Natürlich stellt sich die Frage, warum man eigentlich 4.3 Unix auf einer solchen Maschine betreiben will. Dafür gibt es verschiedene Gründe. Numerisch intensive Programme pflegen nun einmal nicht nur die Dezimalstellen von π zu berechnen, sondern führen eine enorme Zahl an I/O-Vorgängen der verschiedensten Arten aus. Und solche I/Os können nur dann ordentlich unterstützt werden, wenn der Systemkern auf derselben CPU läuft. Die Programmmierer von technisch-wissenschaftlichen Applikationen wünschen sich eine solide Programmierumgebung, und diese ist mit einem Back-End-Prozessor nur schwerlich zu erreichen. Und nicht zuletzt wollten auch die Entwickler ihre Maschine für ihre tägliche Arbeit nutzen, und sie sagten sich: „Wenn selbst wir uns mit unserer eigenen Maschine nicht wohl fühlen, wie können wir dann erwarten, daß ein einsamer Fortran-Programmierer damit arbeiten will?"

Bei der Unix-Implementierung waren Schwierigkeiten und Herausforderungen abzusehen. Dazu gehörte das Interrupt-Handling auf einer Maschine mit vielen Pipelines und hohem Parallelismus, ferner die virtuelle Speicherverwaltung ohne die Unterstützung eines Mikrocodes, Context Switching bei einer hohen Zahl von Registern, das Design eines I/O-Systems für eine Hochleistungsmaschine, deren Ressourcen vom Compiler verwaltet werden. Jedoch auch schon das elementare Problem mit der Größe des Systemcodes bei einer Instruktionsbreite von 256, 512 beziehungsweise 1024 bit und überhaupt der Support des Systemcodes auf einer solchen Architektur stellten große Herausforderungen dar.

Das Resultat der Arbeiten ist, daß die 4.3 BSD Unix-Portierung auf diese Maschine nicht nur gelang, sondern die Ergebnisse außerordentlich ermutigend sind. Es hat sich gezeigt, daß die VLIW-Architektur nicht nur für numerische Probleme hervorragend geeignet ist, sondern daß sich die Vorteile des Feinstrukturparallelismus auch auf den Betriebssystemcode auswirken. ■

Impressum
CHIP PLUS erscheint monatlich als Beilage zu CHIP

Anschrift der Redaktion: Schillerstr. 23a, 8000 München 2, Tel. 0 89-5 14 93-0.
Verlag: Vogel Verlag und Druck KG, Max-Planck-Str. 7/9, 8700 Würzburg, Tel. 09 31-4 18-0.
Redaktionsdirektor: Dr. Gerald O. Dick
Redaktion: Martin Stübs (verantw.), Ulrich Parthier
Grafische Gestaltung: Hans Kuh, Nadja Schubert
Chef vom Dienst: Reiner Lanninger

Textchef: Dr. Alfred Günter Neumann
Autoren dieser Ausgabe: Norbert Vorstädt, Thomas Woywod, Silvia Parthier, Stefan Klusmeier
Verlagsdirektor: Dr. Andreas Kaiser
Anzeigen: Friedrich Mangold
Satz, Litho, Druck, Verarbeitung, Versand: Vogel Verlag und Druck KG

CHIP WISSEN

Die kompetente Buchreihe rund um den PC

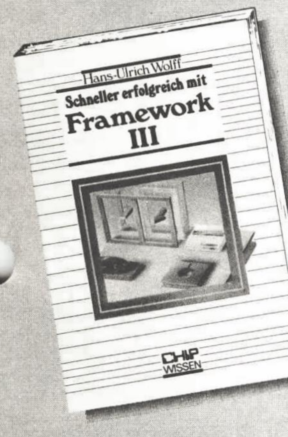


Honerkamp/Jetter:

Fliegen mit dem Flight Simulator III

244 Seiten, zahlr. Bilder, Harteinband,
mit 5,25-Zoll-Mode-Diskette
40,– DM/ISBN 3-8023-**0274**-5

Das Buch beschreibt ausführlich das Flieger-Einmaleins, das zum Beherrschen des Simulationsprogramms **„Flight Simulator III"** notwendig ist. Ausführlich wird die Instrumentenfliegerei mit den dazugehörigen Karten behandelt. Einige Neuerungen, wie der Anflug auf San Francisco Area und Scenery Disk Europe, der Anflüge auf bekannte deutsche und europäische Flughäfen erlaubt, erhöhen den Spaß an diesem Buch.
Die Autoren wünschen schon heute „Hals- und Beinbruch" und „happy landings".



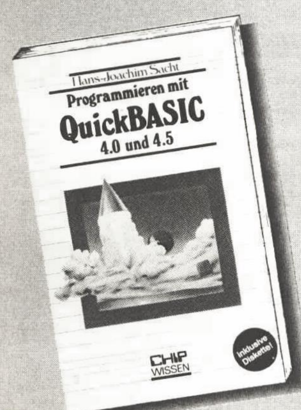
Hans-Ulrich Wolff:

Schneller erfolgreich mit Framework III

180 Seiten, 55 Bilder
Harteinband, Fadenheftung
38,– DM/ISBN 3-8023-**0287**-7

Das Buch beschreibt die deutsche Framework-III-Version. Es erleichtert den Einstieg und unterstützt den Anwender beim Bewältigen der vielfältigsten Aufgaben anhand leicht nachvollziehbarer Beispiele. Die Programmiersprache FRED wird einführend und ihre Funktion umfassend vorgestellt.

Framework III eignet sich hervorragend für die Bewältigung der typischen Aufgabenstellungen am MS-DOS-Rechner: ● konzeptionelles Arbeiten ● Textverarbeitung ● Tabellenkalkulation ● Datenbanken ● Grafik und Telekommunikation



Hans-Joachim Sacht:

Programmieren mit QuickBASIC 4.0 und 4.5

210 Seiten, 66 Bilder, Harteinband
mit 5,25-Zoll-Diskette
48,– DM/ISBN 3-8023-**0245**-1

Durch die neuen Microsoft-Compiler QuickBASIC 4.0 und 4.5 können so komfortable und gutstrukturierte Programme entwickelt werden, wie es sonst nur mit einem leistungsfähigen Interpreter möglich war.

Das Buch bietet die vielfältigen Funktionen in konzentrierter Form und geht vor allem auf die Handhabung und Sprachelemente dieses Compilers ein. Programmierern, die bisher nur mit einem BASIC-Interpreter gearbeitet haben, wird erklärt, wie sie schnell auf QuickBASIC umsteigen und so die Vorteile der Kompilierung ausnutzen können. Eine Diskette mit den im Buch behandelten Beispielen liegt bei.



Hans-Joachim Sacht:

Vom Problem zum Programm

Programmieren in GW-BASIC, Turbo BASIC und QuickBASIC
464 Seiten, 183 Bilder
4., völlig überarbeitete Auflage 1988
48,– DM/ISBN 3-8023-**0214**-1

Das von Microsoft entwickelte GW-BASIC bzw. BASICA gilt inzwischen bei 16-bit-Computern als Standard und für 32-bit-Rechner als Grundlage des Sprachumfangs auch anderer neuer Dialekte.
Die Neukonzeption gliedert sich in 3 Teile:

1. Teil erklärt Grundlagen der BASIC-Programmierung unter Einbeziehung der wichtigsten MS-DOS-Kommandos, Compiler.
2. Teil bringt zahlreiche Programmroutinen für breite Anwendungen.
3. Teil zeigt, wie ein Programm Schritt für Schritt entsteht.

Eine Diskette mit den im Buch behandelten Beispielen ist erhältlich.

VOGEL

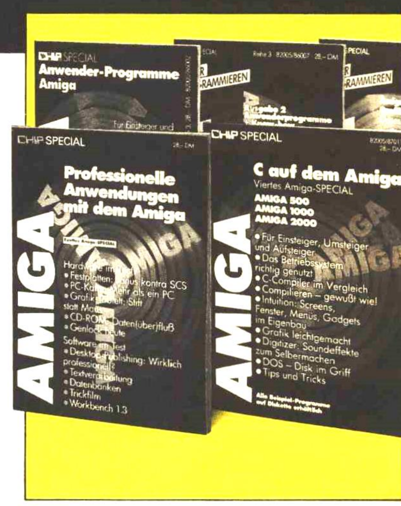
Haben Sie schon den neuen „CHIP- Katalog" ?
Bestellen Sie gleich!

Vogel Buchverlag
Postfach 67 40 · 8700 Würzburg 1

CHIP SPECIAL

AMIGA AMIGA AMIGA

Alles für Ihren AMIGA



AMIGA, Ausgabe 5
Professionelle Anwendungen mit dem Amiga
Was sich mit dem Amiga 2000 bei entsprechender Erweiterung aber auch mit Amiga 500 und 1000 an professionellen Anwendungen realisieren läßt, wurde in langwierigen Testläufen systematisch ermittelt. Die vorliegende Ausgabe enthält getestete Soft- und Hardware-Anwendungen, die professionellen Ansprüchen standhalten: Festplatten, PC-Karte, Grafik-Tablett, CD-ROM, Genlock-Interface, Textverarbeitung, Datenbanken, Trickfilm.

Best.-Nr. 0980, DM/sfr. 28,–, öS 230,–.

AMIGA, Ausgabe 4
C auf dem Amiga
Wo liegen genau die Unterschiede zwischen Amiga 500, 1000 und 2000 und vor allem, ist die Software kompatibel unter den drei Modellen? Mit diesem Special geben die beiden Autoren dem Einsteiger den Schlüssel zu seinem Amiga in die Hand. Die Programmiersprache C eignet sich dazu ausgezeichnet. Die Software für den beschriebenen Amiga-Sounder und alle Programme aus diesem Heft ist für DM 59,– erhältlich. Die Hardware für den Amiga-Sounder kostet DM 134,–

Best.-Nr. 0640, DM/sfr. 28,–, öS 230,–.

AMIGA, Ausgabe 3
Handbuch für Grafikanwendungen
Amiga intern: Speicher und CPU. Die Spezialchips des Amiga. So werden Bilder erzeugt: Grafikfähige Bildschirme; Bitmapping; virtuelle Screens; Playfields; Sprites; Blitter; AnimObjects und AnimComps. Welche Ein- und Ausgabegeräte wofür: Light-Pen; Digitizer; Genlock-Interface; Druckinterface; Drucker; Plotter; Bildschirmfotos. Grafik-Software: Malprogramme; CAD-Programme; Layout-Programme; Animations-Programme; IFF-Files. Im Heft sind u.a. folgende Programme als Listings in AMIGA-BASIC enthalten: Laufschrift; Kalender; Lissalous-Figuren; Funktionsplotter; Farb-Palette; Musik-Demo; Kreisdiagramm; Animation mit Anim-Demo; Mosaik.

Best.-Nr. 0550, DM/sfr. 28,–, öS 230,–.

AMIGA, Ausgabe 2
Anwenderprogramme + Know-How
Aus dem Inhalt: Die Möglichkeiten der Workbench-Version 1.2: Vorstellung der neuen Befehle. Fraktale Welten: Das Mandelbrot-Programm – ein komfortabler Einstieg in die fraktale Geometrie. Im Vergleich: MCC-Pascal gegen LATTICE-C. Business mit VIP-Professional: Die Lotus-kompatible Tabellenkalkulation. Der AMIGA Command Line Interpreter: Wie man den AMIGA ohne Maus bedient. Nützliche Utilities.

Best.-Nr. 0410, DM/sfr. 28,–, öS 230,–.

AMIGA, Ausgabe 1
Anwender -Programme
Die ersten Programme für zwei verschiedene BASIC-Versionen: dem unstrukturierten A-BASIC und dem strukturierten AMIGA-BASIC. Aus dem Inhalt: Profi-Datenbank: Die universelle und frei strukturierbare Datenbank für den AMIGA. NEW: Das nützliche Utility zum Initialisieren Ihres AMIGA-Computers. Ökologie: Ein Programm zum Erlernen der wirtschaftlichen Zusammenhänge. AMIGA-DOS: Menügesteuerter Zugriff in die Tiefen des AMIGA-DOS.

Best.-Nr. 0290, DM/sfr. 28,–, öS 230,–.

Alles für den AMIGA-Fan:

- **Der schnelle C-Einstieg**
  Best.-Nr. 0440, DM/sfr. 28,–, öS 230,–.
- **MIDI-Computer und Musik**
  Best.-Nr. 0966, DM/sfr. 28,–, öS 230,–.
- **Computer steuert Märklin-Eisenbahn**
  Best.-Nr. 0953, DM/sfr. 19,–, öS 150,–.
- **User-Guide zum Prozessor 68000**
  Best.-Nr. 0590, DM/sfr. 28,–, öS 230,–.
- **Programmieren, Ausgabe 3**
  Best.-Nr. 0974, DM/sfr. 19,–, öS 150,–.

Benutzen Sie für Ihre Bestellung bitte die Bestellkarten aus diesem Heft. Für ganz Eilige gibt es den telefonischen Tag- und Nacht-Bestellservice unter der Telefonnummer: 09 31/4 18 22 83.
Weitere Special-Angebote finden Sie in unserem Katalog, den Sie kostenlos beim Verlag anfordern können.

SOFORT BESTELLEN